On Efficient Solution Methods for Mixed-Integer Nonlinear and Mixed-Integer Quadratic Optimization Problems

Von der Universität Bayreuth
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

vorgelegt von

Thomas Lehmann

geboren in Nürnberg

1. Gutachter: Prof. Dr. Klaus Schittkowski
2. Gutachter: Prof. Dr. Jörg Rambau

Tag der Einreichung:
Tag des Kolloquiums:

Fakultät für Mathematik, Physik und Informatik
Angewandte Informatik VII

Abstract[1]

In this thesis we focus on solution methods for convex mixed-integer nonlinear optimization problems (MINLP). As one main result, we propose a new algorithm guaranteeing global optimality for convex MINLPs under standard assumptions. The new algorithm called MIQP-supported outer approximation (MIQPSOA) incorporates the successive solution of convex mixed-integer quadratic programs (MIQP) in a linear outer approximation framework. An extensive numerical competitive study based on several different MINLP solvers shows, that a first implementation of the new method performs well in terms of both the reliability and the efficiency. Since the new method is designed to solve simulation-based optimization problems arising in practical engineering applications, the main performance criterion is the number of function evaluations required to solve a problem. Furthermore, the test results indicate, that the integration of mixed-integer search steps, resulting from the solution of convex MIQPs, significantly improves the reliability and the efficiency compared to well-known linear outer approximation methods.

After reviewing available solution techniques for convex MINLP problems, we present the algorithmic set-up as well as the convergence proof of MIQPSOA. As pointed out in this dissertation, MIQPSOA is a first step towards a convergent MINLP solution method, that solely relies on the successive solution of convex MIQPs as proposed by Exler and Schittkowski. Finally, we present an extensive numerical test case study considering different solution methods for convex MINLPs.

The second part of this thesis deals with efficient solution techniques for convex mixed-integer quadratic programs, that arise as subproblems during the solution of MINLPs by MIQP-based algorithms, such as MIQPSOA. First, we briefly review latest developments in state-of-the-art mixed-integer linear (MILP) solvers, since we want to develop a MIQP solver that incorporates the most successful components of MILP solvers. As we focus on branch-and-bound methods, one main component is an efficient and robust sub-solver for continuous quadratic programs, which is able to perform warmstarts. On the other hand, cutting planes have led to a tremendous speed-up of mixed-integer linear solvers during the last 20 years. As a consequence, we extend an efficient construction method for disjunctive cutting planes, such that it can be applied for MIQPs.

Extensive numerical tests show, that the performance of a branch-and-bound solver can be significantly increased by exploiting warmstarts. Furthermore, it turns out, that in a majority of the test cases, where disjunctive cutting planes exist, the calculation times are reduced up to a factor of more than 5. Nevertheless there are also instances, where the presents of disjunctive cutting planes significantly slows down the performance. Due to the efficient cut generation method developed within this thesis, the generation of cutting planes has almost no influence on the calculation time, if no disjunctive cuts exist, which is the case in about 45 % of all test instances. As a

[1] The research was supported by Shell SIEP Rijswijk, GameChanger Project IDC-2005050006, SIEP-EPR-RIR

consequence, the application of cutting planes for MIQPs needs further attention and especially a dynamic cut management might be very profitable. Finally, we compare the performance of our branch-and-cut solver MIQL with the solver SCIP, which is one of the state-of-the-art MILP solvers, that can also solve MIQPs. These tests indicate, that MIQL outperforms SCIP on hard MIQP instances, while SCIP is faster for simpler test cases.

Zusammenfassung[2]

Im Fokus dieser Dissertation stehen Optimierungsverfahren für konvexe, gemischt-ganzzahlige, nichtlineare Optimierungsprobleme (MINLP). Ein wesentliches Resultat dieser Arbeit ist die Entwicklung eines neuen Lösungsverfahren. Dieser Algorithmus, der MIQP-supported Outer Approximation (MIQPSOA) genannt wird, garantiert globale Optimalität der Lösung unter üblichen Voraussetzungen. MIQPSOA basiert auf der sukzessiven Lösung von konvexen, gemischt-ganzzahligen, quadratischen Teilproblemen (MIQP) innerhalb eines Linear Outer Approximation Ansatzes. Eine ausführliche numerische Studie mehrerer verschiedener MINLP-Lösungsverfahren zeigt, dass die erste Implementierung der neuen Methode sehr effizient und gleichzeitig robust ist. Da wir hauptsächlich simulations-basierte Optimierungsprobleme aus Anwendungen des Ingenieurswesens lösen, stellt die Anzahl der innerhalb des Lösungsprozesses benötigten Funktionsauswertungen das wichtigste Performance-Kriterium dar. Die numerische Analyse zeigt, dass die Integration von gemischt-ganzzahligen Suchschritten, die aus der Lösung der MIQP-Teilprobleme bestimmt werden, sowohl die Robustheit als auch die Effizienz im Vergleich zum bekannten Verfahren der Linear Outer Approximation deutlich verbessert.
Nachdem wir bekannte Verfahren zur Lösung von konvexen, gemischt-ganzzahligen, nichtlinearen Optimierungsproblemen vorgestellt haben, wird der Algorithmus MIQP-SOA motiviert und beschrieben. Anschließend werden seine Konvergenzeigenschaften untersucht und Konvergenz für konvexe MINLPs bewiesen. Außerdem werden zukünftige Verfahren skizziert, die Konvergenz für konvexe Probleme garantieren könnten und allein auf der sukzessiven Lösung von konvexen MIQPs basieren. Abschließend vergleichen wir die Performance und die Robustheit von verschiedenen MINLP Optimierungsverfahren.
Der zweite Teil der Dissertation beschäftigt sich mit effizienten Lösungsverfahren für konvexe, gemischt-ganzzahlige, quadratische Optimierungsprobleme, die als Teilprobleme in MIQP-basierten Lösungsverfahren, wie beispielsweise MIQPSOA, auftreten. Ausgehend von einer kurzen Zusammenfassung der wichtigsten Entwicklungen bei aktuellen Lösungsverfahren für gemischt-ganzzahlig lineare Optimierungsprobleme (MILP), wird ein MIQP-Löser entwickelt. Dieser Löser namens MIQL beruht auf den wesentlichen Komponenten von aktuellen MILP-Lösungsverfahren. Da es sich bei MIQL um ein QP-basiertes Branch-and-bound Verfahren handelt, ist der effiziente und robuste Teilproblem-Löser für kontinuierliche quadratische Probleme von entscheidender Bedeutung. Besonders wichtig ist dabei die Fähigkeit, verwandte Probleme schnell mit sogenannten Warmstart-Methoden lösen zu können. Ein weiteres Hauptaugenmerk dieser Arbeit ist die Verwendung von allgemeinen Schnittebenen-Verfahren für nicht-basis Lösungen, wie sie bei der kontinuierlichen Relaxierung der MIQPs auftreten. Die Anwendung von Schnittebenen hat bei MILP-Lösern zu einer erheblichen Steigerung der Leistungsfähigkeit geführt, weshalb in dieser Dissertation erst-

[2] The research was supported by Shell SIEP Rijswijk, GameChanger Project IDC-2005050006, SIEP-EPR-RIR

malig effiziente Konstruktionsverfahren für Schnittebenen entwickelt werden, die auch für MIQPs angewendet werden können.
Eine ausführliche numerische Analyse zeigt, dass die Performance des Branch-and-bound Verfahrens durch Warmstarts signifikant verbessert werden kann. Weiterhin kann man feststellen, dass in der Mehrzahl der Fälle, in denen Schnittebenen des Types "disjunctive cutting planes" existieren, die Rechenzeiten deutlich, bis zu einem Faktor von mehr als 5, reduziert werden können. Trotzdem gibt es auch vereinzelte Instanzen, bei denen die Rechenzeit durch die Integration von disjunctive cutting planes stark erhöht wird. Das in dieser Dissertation entwickelte Schnittebenen-Verfahren selbst beeinflusst die Rechenzeit kaum. Dies ist besonders dann von Vorteil, falls keine disjuncitve cutting planes existieren, was bei der verwendeten Testbibliothek in circa 45 Prozent der Probleme der Fall ist.
Zusammenfassend bleibt festzustellen, dass die Verwendung von Schnittebenen-Verfahren für MIQPs weitere Forschung notwendig macht und besonderes ein dynamisches Schnitt-ebenen-Management sehr profitabel erscheint. Im Vergleich zum Löser SCIP, der zu den besten Lösern für MILPs gehört und auch MIQPs lösen kann, stellt sich heraus, dass MIQL gerade bei schwierigen MIQP-Problemen deutlich überlegen ist.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Mixed-integer nonlinear programming (MINLP) is a challenging optimization area, since it combines nonlinear programming and mixed-integer programming. Although there exists lots of different applications, only few efficient solution methods are developed yet and most of them are hardly applicable for practical problems relying on complex simulation software. This thesis is focused on the development and the implementation of algorithms for solving mixed-integer nonlinear optimization problems arising in industrial engineering applications.

In mixed-integer nonlinear programming a nonlinear scalar objective function is minimized subject to nonlinear equality and inequality constraints,

$$\begin{aligned} & x \in X,\ y \in Y: \\ & \quad \begin{aligned} \min \quad & f(x,y) \\ \text{s.t.} \quad & g_j(x,y) = 0, \quad j = 1,\dots,m_e, \\ & g_j(x,y) \geq 0, \quad j = m_e+1,\dots,m. \end{aligned} \end{aligned} \tag{1.1}$$

The number of equality constraints is denoted by m_e, while m is the total number of constraints. The two sets X and Y are defined by lower bounds $x_l \in \mathbb{R}^{n_c}$, $y_l \in \mathbb{N}^{n_i}$ and upper bounds $x_u \in \mathbb{R}^{n_c}$, $y_u \in \mathbb{N}^{n_i}$ on continuous and integer variables,

$$\begin{aligned} X &:= \{x \in \mathbb{R}^{n_c} : x_l \leq x \leq x_u\}, \\ Y &:= \{y \in \mathbb{N}^{n_i} : y_l \leq y \leq y_u\}, \end{aligned} \tag{1.2}$$

where n_c is the number of continuous variables and n_i is the number of integer variables. The total number of variables is denoted by

$$n := n_c + n_i. \tag{1.3}$$

All model functions may depend on continuous variables $x \in \mathbb{R}^{n_c}$ and integer variables $y \in \mathbb{N}^{n_i}$. Note, that binary variables are also contained in Y, i.e., $y_i \in \mathbb{B} := \{0,1\}$ for some $i \in I := \{1,\dots,n_i\}$. I denotes the index set of integer and binary variables. The index set of the continuous variables is given by J, while the index set of the constraints of MINLP (1.1) is defined by

$$\mathbb{J} := \mathbb{J}_= \cup \mathbb{J}_> = \{1,\dots,m\}, \tag{1.4}$$

with

$$\begin{aligned} \mathbb{J}_= &:= \{1,\dots,m_e\}, \\ \mathbb{J}_> &:= \{m_e+1,\dots,m\}. \end{aligned} \tag{1.5}$$

It is assumed that the functions $f(x,y)$ and $g_j(x,y)$, $j = 1,\dots,m$, are twice continuously differentiable subject to all $x \in X$. We consider general problems and do not restrict our research to specific formulations, e.g., problems where $f(x,y)$ and $g(x,y) := (g_1(x,y),\dots,g_m(x,y))^T$ depend linearly on the integer variables $y \in Y$. A linear relationship between integer variables and problem functions is often assumed and exploited by specifically tailored algorithms, see, e.g., Floudas [52].

An important subclass of problem (1.1) are convex mixed-integer nonlinear optimization problems given by

$$\begin{aligned} & x \in X,\ y \in Y: \\ & \qquad \min \quad f(x,y) \\ & \qquad \text{s.t.} \quad g_j(x,y) \ \geq \ 0, \quad j \in \mathbb{J}_{>}. \end{aligned} \tag{1.6}$$

Problem (1.6) may in addition contain m_e linear equality constraints given by

$$a_j^T \begin{pmatrix} x \\ y \end{pmatrix} - b_j \ = \ 0, \quad j \in \mathbb{J}_{=}, \tag{1.7}$$

with $a_j \in \mathbb{R}^n$, $j \in \mathbb{J}_{=}$ and $b_j \in \mathbb{R}$, $j \in \mathbb{J}_{=}$, but they are omitted in the sequel to improve the readability. Problem (1.6) differs from the general mixed-integer nonlinear program (1.1), since it possesses a convex feasible region and a convex objective function. This means, that the inequality constraints $g_j(x,y)$, $j \in \mathbb{J}_{>}$ are concave on $X \times Y_{\mathbb{R}}$ while the objective function is convex on $X \times Y_{\mathbb{R}}$ with

$$Y_{\mathbb{R}} := \{y \in \mathbb{R}^{n_i} : y_l \leq y \leq y_u\}, \tag{1.8}$$

i.e., the set $Y_{\mathbb{R}}$ is the continuous relaxation of Y. Furthermore all functions are twice differentiable subject to $x \in X$ and $y \in Y_{\mathbb{R}}$. In general, a continuous relaxation of a mixed-integer program is a relaxation of the original problem, where integer variables are replaced by continuous variables, i.e., the domain of the integer variables Y is relaxed by the set $Y_{\mathbb{R}}$.

There exists a variety of different algorithms for solving the convex mixed-integer nonlinear optimization problem (1.6) to global optimality, see, e.g., Floudas [52], Grossmann and Kravanja [61] or Bonami, Kilinc and Linderoth [28] for review papers. This thesis focuses on the development of solution methods for general convex MINLP problems based on the successive solution of mixed-integer quadratic subproblems. Furthermore, the proposed algorithm should successfully be applied to solve simulation-based optimization problems arising in engineering applications. The resulting algorithm, called MIQP-supported outer approximation (MIQPSOA), as well as the solution methods, that are reviewed in Chapter 2, rely on gradient information.

The general mixed-integer nonlinear program (1.1) or the convex counterpart (1.6) can be tackled by a variety of different algorithms. In general, one can distinguish between heuristic approaches and global optimization techniques. Heuristic methods can

not guarantee global optimality. Furthermore, global optimization techniques can be applied heuristically, if some requirements are not satisfied, e.g., applying a method that guarantees global optimality for a convex MINLP (1.6) to solve a non-convex MINLP (1.1). Other heuristic approaches are stochastic search methods such as pattern search algorithms searching the integer space, see Audet and Dennis [7].

If MINLP (1.1) is explicitly given in algebraic form, global optimization techniques can be applied, see, e.g., Couenne developed by Belotti et al. [21]. It is also possible to replace the integrality condition by continuous nonlinear constraints and to solve the resulting highly non-convex program by continuous global optimization algorithms, see, e.g., Li and Chou [77]. Nowak et al. [82] derive lower bounds by linear outer approximations, while upper bounds are provided by local nonlinear programming (NLP) solution techniques. The quality of the upper and lower bound is successively improved by spatial branching. Often specialized exact solution methods or heuristics are tailored for specific MINLP problem instances. These techniques can be very efficient but their application is restricted, see e.g., Möller [81] or Fügenschuh et al. [55].

In practice, the requirements of available solution methods are often violated. Many simulation-based mixed-integer nonlinear problems are not relaxable and model functions are often highly non-convex. A practical situation is considered by Bünner, Schittkowski and van de Braak [33], where typical integer variables are the number of fingers and layers of an electrical filter. Moreover, some of these approaches require detection of infeasibility of continuous nonlinear programs, which is highly unattractive from the computational point of view.

Within this thesis a new algorithm for solving convex mixed-integer nonlinear optimization problems is developed. In contrast to available solution methods, it is based on the successive solution of strictly convex mixed-integer quadratic programming (MIQP) subproblems. The motivation is to extend the well-known concept of sequential quadratic programming (SQP), which is one of the most widely used solution technique for practical continuous nonlinear optimization methods. A corresponding algorithm whose implementation is called Mixed-Integer Sequential Quadratic Programming (MISQP) has already been proposed by Exler and Schittkowski [45]. It turns out, that it can very efficiently be applied to solve simulation-based mixed-integer optimization problems, see Chapter 6, but no convergence proof even for convex MINLPs (1.6) could be given.

Mixed-integer quadratic programming has not received much attention in optimization in the past. At present a number of researchers starts to focus on mixed-integer quadratic programming, see e.g., Buchheim, Caprara and Lodi [32]. Nevertheless, only few theory and some software is available, e.g., the solver SCIP developed by Achterberg [2] and the solver CPLEX, developed by IBM/ILOG [41]. This is especially the case, if we consider the MIQP formulation with a quadratic objective function and linear constraints, see (1.9), which is closely related to mixed-integer linear programming (MILP). One reason is that mixed-integer linear solvers are very powerful

nowadays, which implies that MILPs are set up whenever possible. An application of mixed-integer quadratic programming is portfolio optimization, see Bienstock [25]. Since MISQP, see Section 2.10, and MIQPSOA, proposed in Section 3.1, are based on the successive solution of strictly convex mixed-integer quadratic programs (MIQP), fast and robust algorithms are required for solving these problems. Every speed-up obtained for the solution of MIQP problems directly increases the efficiency of the MIQP-based MINLP solvers.

We consider the mixed-integer quadratic program possessing a strictly convex objective function restricted by linear equality and inequality constraints:

$$\begin{aligned} & x \in X,\ y \in Y: \\ & \begin{aligned} \min \quad & \frac{1}{2}\left(x^T, y^T\right) B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix} \\ \text{s.t.} \quad & A_E \begin{pmatrix} x \\ y \end{pmatrix} = b_E, \\ & A_I \begin{pmatrix} x \\ y \end{pmatrix} \geq b_I. \end{aligned} \end{aligned} \tag{1.9}$$

x and y denote the vectors of the continuous and integer variables, respectively, while $B \in \mathbb{R}^{n \times n}$ is a positive definite matrix and $c \in \mathbb{R}^n$ holds. X and Y are defined by the upper and lower bounds on both the continuous and the integer variables, see (1.2). n_c denotes the number of continuous variables and n_i is the number of integer variables. The total number of variables is denoted by n, i.e., $n := n_i + n_c$. Equality constraints are denoted by $A_E \in \mathbb{R}^{m_e \times n}$ and $b_E \in \mathbb{R}^{m_e}$, while inequality constraints are given by $A_I \in \mathbb{R}^{m_i \times n}$ and $b_I \in \mathbb{R}^{m_i}$. Therefore m_e denotes the number of equality constraints, while m_i is the number of inequality constraints.

An appropriate solution approach for MIQP (1.9) is branch-and-bound, which is a general concept to solve optimization problems, see Dakin [42] and Section 2.4. Branch-and-bound is especially applied for solving mixed-integer optimization problems, but it also plays a crucial role in global optimization of continuous non-convex nonlinear programming problems, see Tawarmalani and Sahinidis [99]. Section 2.4 provides a detailed introduction on the general branch-and-bound concept. For mixed-integer optimization problems a branch-and-bound method usually starts by solving the continuous relaxation of the original problem, i.e., the set Y is replaced by $Y_{\mathbb{R}}$. We denote the optimal solution of the continuous relaxation by $(\bar{x}, \bar{y}) \in X \times Y_{\mathbb{R}}$, while the optimal solution of the original problem is given by $(x^*, y^*) \in X \times Y$. Since $(\bar{x}, \bar{y})$ is the optimal solution of a relaxation, the corresponding value of the objective provides a lower bound on the optimal objective value at the solution (x^*, y^*). If $\bar{y} \in Y$ holds, $(\bar{x}, \bar{y})$ is the optimal solution of the mixed-integer optimization problem. Otherwise, the branch-and-bound method commonly continues by choosing an integer variables y_i possessing a fractional value at $\bar{y}$, i.e., $\bar{y}_i \notin \mathbb{N}$. Considering $\bar{y}_i$ the problem can be partitioned into two disjoint problems with more stringent bounds derived from $\bar{y}_i$, i.e., $y_i \leq \lfloor \bar{y}_i \rfloor$ and $y_i \geq \lceil \bar{y}_i \rceil$. Repeating this process leads to an enumeration of the integer search space, which terminates as soon as the optimal solution is found

and its optimality is proved. This enumeration process can be illustrated by a search tree. The nodes commonly correspond to relaxations of the partitioned subproblems. Applying a branch-and-bound method for solving strictly convex MIQP yields strictly convex quadratic relaxations, that can be solved efficiently by continuous quadratic programming (QP) solvers.

Strictly convex mixed-integer quadratic programming is a special case of convex mixed-integer nonlinear programming. As a consequence, a convex MIQP can also be solved by other algorithms apart from branch-and-bound methods. Some of these techniques are presented in Chapter 2. These methods often rely on linear relaxations of nonlinear problem functions, that are iteratively refined during the optimization process. Since the feasible region of a MIQP is already described by linear constraints, these approaches work on the exact feasible region and no iterative refinement subject to the constraints is necessary. In contrast to branch-and-bound methods, these solution techniques allow the application of powerful mixed-integer linear solvers, on the cost of solving a hopefully small sequence of mixed-integer linear problems, see e.g., Section 2.5 for further details.

The branch-and-bound approach is the classical way to solve mixed-integer programs. All state-of-the-art solvers for mixed-integer linear programming (MILP) rely on the branch-and-bound concept. But MILP solvers contain lots of additional components increasing the performance of the branch-and-bound enumeration. The integration of these techniques led to a huge progress and turned mixed-integer linear solvers into very powerful algorithms. Their tremendous improvement is based on three major components, as presented by Bixby for the solver CPLEX of IBM/ILOG [41]:

- Powerful continuous linear solvers possessing excellent warmstart features. When performing a warmstart, an algorithm exploits information of previous runs on similar problems, which can significantly reduce the computational effort.

- A large variety of cut generators allowing the construction of deep cutting planes, tightening the feasible region of the continuous relaxation.

- Advanced presolve techniques reducing the problem complexity prior to the branch-and-bound enumeration process.

Apart from these three techniques other components helped to increase the power of state-of-the-art mixed-integer linear solvers, e.g., heuristics. The largest improvement is caused by the generation of cutting planes, see Table 1.1, which shows the improvements caused by different techniques. Solution methods combining branch-and-bound and the generation of cutting planes are called branch-and-cut solvers.

Technique	Factor of Speed-up
Cutting planes	53.7
Presolve	10.8
Branching rules	2.9
Heuristics	1.4
Node presolve	1.3
Probing on dives	1.1

Tab. 1.1: Speed-up Factors for Mixed-Integer Linear Programming, Bixby [26]

MIQPs and MILPs only differ in the objective function, since both possess a polyhedral feasible region. As a consequence, all mixed-integer linear techniques, that are independent of the objective function, can be applied directly for MIQPs as well. Especially, all presolving procedures, where the objective function has no influence can be used in a MIQP solver.

Nevertheless, the solution of MIQPs by the branch-and-bound approach does hardly profit from these developments. The main difference between NLP-based branch-and-bound for solving MINLPs and QP-based branch-and-bound for solving strictly convex MIQPs, is the reduced effort needed to solve a QP instead of a NLP.

One major task of this dissertation is to develop a branch-and-cut mixed-integer quadratic solver, that is influenced by the improvements of MILP solvers. By incorporating mixed-integer linear techniques, we want to improve the performance significantly compared to pure branch-and-bound methods.

In general, a branch-and-cut method consists of three algorithmic components. Obviously, an enumeration routine is required to carry out the branch-and-bound process. Furthermore, the subproblems corresponding to the nodes of the constructed search tree need to be solved. Therefore, the second component is a QP solver possessing appropriate warmstart features for solving the continuous relaxations. The last component are cut generators, which tighten the problem formulation of the quadratic relaxation. The generation of cutting planes for MIQP problems is a major task of this thesis.

In mixed-integer linear programming, cutting planes have been studied since a famous paper of Gomory [59] in 1958. Traditionally, a cutting plane is a linear inequality, that cuts off the solution of a relaxation of the original problem, while retaining all feasible integer solutions. Therefore, cutting planes truncate the feasible region of the relaxation and leave the feasible region of the original problem unchanged. As a consequence, they lead to an improved lower bound given by the relaxation. Considering general mixed-integer nonlinear programming, cutting planes can be used to solve convex MINLP problems, see Section 2.7.

Although cutting planes have been applied very successfully within branch-and-cut solvers for mixed-integer linear programming, there are almost no results on the gener-

ation of cutting planes for mixed-integer quadratic programming. That's why cutting planes are analyzed in the context of mixed-integer quadratic programming problems, in order to develop a MIQP branch-and-cut solver.

All our technical expertise obtained in the last couple of years is implemented in the mixed-integer quadratic solver MIQL. Implementation details and a definition of the calling parameters can be found in Lehmann et al. [71].

There are plenty of different applications leading to mixed-integer nonlinear models. Since this research was supported by the company Shell, we focus on applications arising in petroleum industry. In other areas many applications lead to MINLP problems, see Floudas [52]. The reason is that the problem class MINLP provides lots of freedom to describe difficult interactions. The bottleneck is the lack of efficient solution methods.

In this thesis we gain insights into mixed-integer nonlinear optimization problems arising in petroleum industry. We present briefly three different realistic applications that lead to a variety of test cases: lift gas distribution, well relinking, and the Sarawak SGPS model, see Barton and Selot [97]. The complexity of the corresponding models varies from easy-to-solve *toy* problems to extremely complex applications.

The output of an oil field is increased significantly by injecting lift gas, see e.g., Wang and Litvak [106]. The goal is to maximize the total oil production subject to a given limited amount of lift gas. Lift gas is typically injected into different wells and the major difficulty is the so-called non-instantaneous response of some wells. This means that the oil production is only increased, if a certain threshold of injected lift gas is exceeded. Otherwise the lift gas is wasted.

Some of these non-instantaneous wells become active at the optimal solution, i.e., need to be provided with a certain amount of lift gas exceeding the corresponding threshold. As a consequence, non-instantaneous wells cannot be ignored. If the lift gas model is set up as a continuous nonlinear problem, gradient-based solvers easily run into difficulties, since the gradient of the model functions for non-instantaneous wells is zero until the threshold is exceeded. To avoid this situation the corresponding models can be extended by introducing binary variables to turn on and off non-instantaneous wells. As a consequence, the problem class changes from NLP to MINLP and we obtain only sensible gradient information.

Since the output of an oil field should be maximized subject to the injection of lift gas, one function evaluation corresponds to a simulation of the response of the oil field. It cannot be expected that the injection leads to an immediate increase of the oil production and therefore the simulation has to comprise some years. Thus, function evaluations are very time-consuming for this application.

To be able to deal with this application without having to install the simulation tools, a simplified, analytic problem formulation for lift gas optimization can be modeled in different ways. The lift gas curves for each well describe the response of the oil field depending on the amount of lift gas, that is injected. They can be constructed

by linearly interpolating table data. Thus, the corresponding model functions are not differentiable and violate a basic assumption of available solution methods, see Chapter 2. Replacing these non-differentiable functions by fitting differentiable functions overcomes this drawback. Nevertheless, this modeling is inappropriate, since the corresponding gradients are zero on large areas inside the domain of the variables. Therefore, every gradient-based solution method does not gain sufficient information, within such an area. As a consequence, progress towards an optimal solution can hardly be obtained by any gradient-based method. It is possible to model the main characteristics of lift gas optimization problems by a convex MINLP. Therefore, convergence towards the global optimum is guaranteed by NLP-based branch-and-bound or other methods, see Chapter 2. Nevertheless, such methods are very expensive in terms of the number of function evaluations, see Chapter 6.

Well relinking problems are non-convex mixed-integer nonlinear optimization problems, where the total oil flow in a network is to be maximized. The network consists of a given number of source nodes and a number of sink nodes. Each source node has to be connected to exactly one sink node, while the total capacity at the sinks is limited in terms of pressure and flow. Each source node has a nonlinear pressure-flow characteristic, while the total flow within the network is bounded. Figure 1.1 illustrates source nodes on the left and sink nodes on the right hand side. Every source is connected to each sink. So-called network conditions ensure that exactly one connection is active for each source node.

For the first variant of well relinking models, sinks are disconnected. In an extended second model, compressors are included, which allow connections between sinks. The compressors possess limited resources and their configuration is part of the optimization. In general, pressure and flow depend on each other. The higher the pressure, the smaller is the corresponding flow and vice versa. This relation is called well performance curve and typically varies among the sources which are also called wells. The pressure flow interactions are modeled by nonlinear functions, whereas the network is represented by binary variables and split factor constraints to ensure the required network conditions.

The Sarawak SGPS model is a large non-convex mixed-integer nonlinear program to model the upstream gas production system, see Barton and Selot [97]. The model includes a multi-product network, nonlinear pressure-flow rate relationships, production-sharing contracts and operational rules. It was inspired by the Sarawak gas production system (SGPS), which consists of 12 offshore fields and three associated gas fields. The optimal routing of gas within the network has to ensure a given level of quality at the liquefied natural gas (LNG) plants, see Figure 1.2. The network is controlled by regulating pressures.

The SGPS is operated by a single operating company, although several parties own the fields and the LNG plants. Thus, the operator has to consider a complex system of production-sharing contracts, which may even prohibit the supply of a certain field to a specified plant.

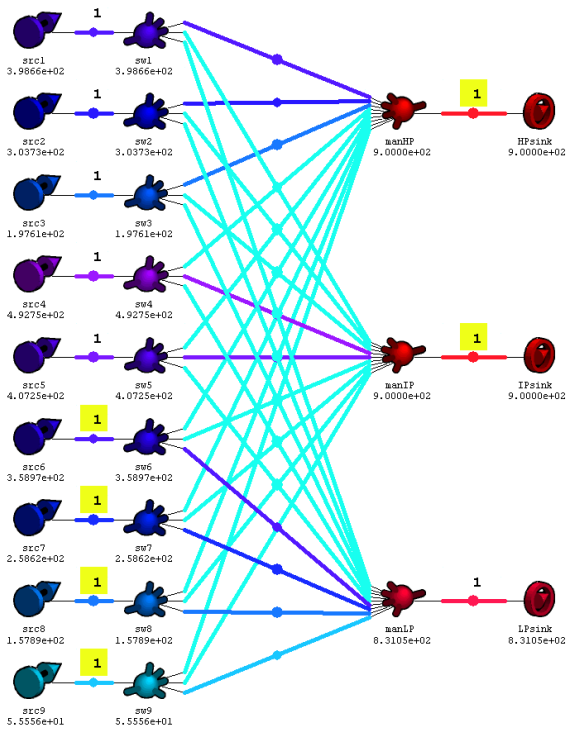

Fig. 1.1: Graphical Representation of a Well Relinking Network

The model is designed to plan optimal steady-state operations over a short term period, i.e., two to twelve weeks. It consists of two interacting submodels. The first one considers the physical constraints associated with the current production network and facilities. It is called infrastructure model. The second one models consumer requirements, operational rules, contractual obligations, and is called contractual rule model.

The resulting MINLP problem consists of 827 variables including 23 binary variables, where 702 of the total 1094 constraints are equations. It is implemented in the modeling language GAMS, see Brooke et al. [31]. In Barton and Selot [19] it is claimed that the production rate can be increased by around 2.5 %, which is equivalent to an increase of 60-70 million dollar in annual revenue.

This thesis is subdivided as follows. Chapter 2 provides a brief review on well-known theory of continuous nonlinear programming, see Section 2.1 and on available methods for both continuous and convex mixed-integer nonlinear optimization problems, see Section 2.2 to Section 2.8. Section 2.9 reviews an innovative approach of Leyffer [76] based on the work of Borchers and Mitchell [29] to integrate the solution of continuous nonlinear subproblems, which arise during NLP-based branch-and-bound, in the corresponding enumeration scheme. The research presented in this thesis is motivated by an innovative solution approach for MINLPs derived from nonlinear

programming techniques. The corresponding algorithm is an extension of a sequential quadratic programming (SQP) trust region method and is proposed by Exler and Schittkowski [45]. The corresponding implementation is called MISQP, which is described in Section 2.10. In contrast to most other available solution techniques, it can be applied under realistic conditions enforced by simulation-based mixed-integer nonlinear optimization problems. Extensive numerical tests show, that this approach yields good results for academic test cases and practical applications in petroleum industry. Furthermore, we present the basic concepts of a branch-and-bound solution method for strictly convex mixed-integer quadratic optimization problems in Section 2.11. It contains a brief description of a continuous quadratic solver and analyses the possibility of performing warmstarts during branch-and-bound.

Encouraged by the promising numerical results obtained by MISQP and the absence of a convergence proof, a new algorithm called MIQPSOA is developed in Chapter 3. Based on the successive solution of strictly convex mixed-integer quadratic subproblems, it guarantees convergence properties for convex mixed-integer nonlinear programs. The algorithm is motivated and described in Section 3.1. The convergence analysis is carried out in Section 3.2 and the direction of future research towards pure MIQP-based techniques is pointed out in Section 3.3 together with some implementation aspects.

In Chapter 4 we introduce some basic theory on linear programming. Furthermore, we review the basic concept of disjunctive programming and the related disjunctive cutting planes. The main focus is the efficient generation of disjunctive cutting planes for MILPs proposed by Balas and Perregaard [17].

Chapter 5 subsumes our research on cutting planes for MIQPs. In Section 5.1 we focus on disjunctive cutting planes, that can be constructed for non-basic solutions, which correspond to solutions of the continuous relaxation of a MIQP. Since the standard construction method is very expensive, we extend the efficient generation method for basic solutions to non-basic solutions, see Sections 5.2 and 5.3. This yields an efficient cut generator for MIQPs.

Apart from theoretical analysis, we focus on the development of software. It is supposed to be able to tackle simulation-based optimization problems arising in engineering applications. We present numerical performance data for different solution methods, which are based on the implementation of the proposed theoretical concepts. The results are obtained for a large number of both academic and engineering test cases and are presented in Chapter 6.

Note, that we always consider minimization problems unless stated otherwise.

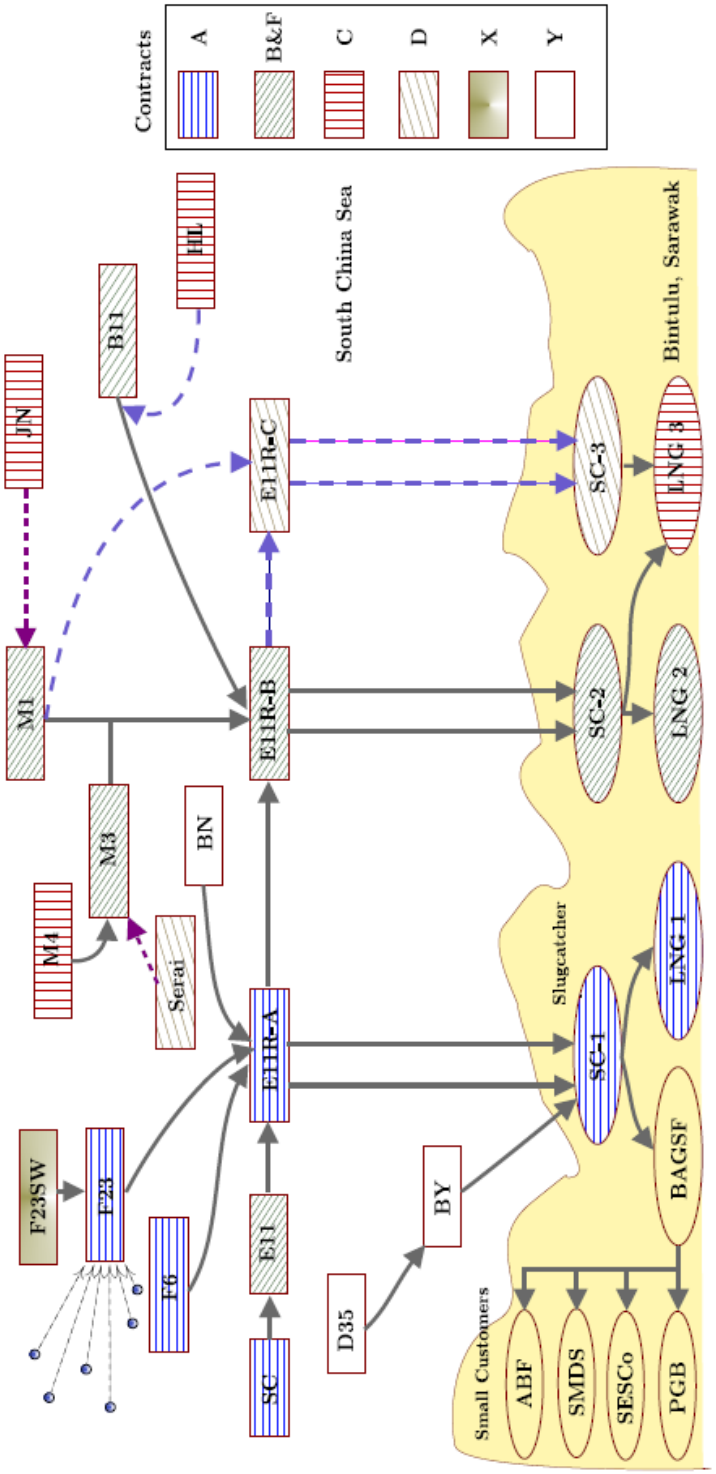

Fig. 1.2: The Savarak Gas Production System, Barton and Selot [97]

2. AVAILABLE NONLINEAR SOLUTION TECHNIQUES

One major goal of this thesis is the development of new solution methods for convex mixed-integer nonlinear optimization problems. The basis are the well-known and widely established sequential quadratic programming methods (SQP). They are to be extended for mixed-integer optimization problems, such that global optimality is guaranteed for convex MINLPs (1.6). In this chapter, we introduce some basic theory on nonlinear programming (NLP) and review the trust region method of Yuan [112], which can solve NLP problems efficiently. Furthermore, we give an overview on existing solution techniques for convex mixed-integer nonlinear optimization problems. Furthermore, we present an innovative approach proposed by Exler and Schittkowski [45], which motivates the development of a new solution method called MIQPSOA, presented in Chapter 3. Finally, we present some of your work on the solution of convex MIQPs, focusing on warmstarts.

2.1 Basic Theory of Continuous Nonlinear Optimization

In this section, we provide a brief review on basic theory of continuous nonlinear programming. All other sections, dealing with nonlinear programming will refer to these basic definitions and theorems. We consider the general continuous nonlinear program given by

$$\begin{aligned} & x \in \mathbb{R}^n : \\ & \qquad \begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_j(x) = 0, \quad j = 1, \ldots, m_e, \\ & g_j(x) \geq 0, \quad j = m_e + 1, \ldots, m. \end{aligned} \end{aligned} \tag{2.1}$$

The continuous nonlinear optimization problem (2.1) is obtained from MINLP (1.1), if the integer variables $y \in \mathbb{N}^{n_i}$ are omitted or relaxed, i.e., $n_i = 0$ and $n = n_c$. The index set of the constraints of NLP (2.1) is defined by

$$\mathbb{J} := \mathbb{J}_= \cup \mathbb{J}_> = \{1, \ldots, m\} \tag{2.2}$$

with

$$\begin{aligned} \mathbb{J}_{=} &:= \{1, \ldots, m_e\}, \\ \mathbb{J}_{>} &:= \{m_e + 1, \ldots, m\}. \end{aligned} \tag{2.3}$$

Furthermore, the bounds on the continuous variables introduced in (1.2) are integrated in the constraints $g_j,\ j \in \mathbb{J}$. This means, that the original nonlinear constraints, for the moment denoted by $\tilde{m}$, are extended by n upper and n lower bounds on the continuous variables, i.e.,

$$\begin{aligned} g_{\tilde{m}+i}(x) &:= -x_i + e_i^T x_u \geq 0, \quad \forall i \in \{1, \ldots, n\}, \\ g_{\tilde{m}+n+i}(x) &:= x_i - e_i^T x_l \geq 0, \quad \forall i \in \{1, \ldots, n\}, \end{aligned} \tag{2.4}$$

where e_i denotes the i-th unit vector. Nevertheless, we denote the feasible domain induced by the bounds on the continuous variables by X, i.e.,

$$X := \{x \in \mathbb{R}^n : g_{\tilde{m}+i}(x) \geq 0,\ i \in \{1, \ldots, 2n\}\}. \tag{2.5}$$

To be consistent with standard notation, we define the number of constraints to be $m := \tilde{m} + 2n$ and extend the index set $\mathbb{J}_{>}$ accordingly. m_e still denotes the number of equality constraints.

The objective function $f(x)$ and the constraints $g_j(x),\ j = 1, \ldots, m$, are twice continuously differentiable subject to $x \in X$. Moreover, the feasible region is given by Definition 2.1.

Definition 2.1. *The feasible region of NLP* (2.1) *is defined by the set*

$$\begin{aligned} \mathbb{F} := \quad & \{x \in \mathbb{R}^n :\ g_j(x) = 0,\ j = 1, \ldots, m_e\} \\ & \cap \{x \in \mathbb{R}^n :\ g_j(x) \geq 0,\ j = m_e + 1, \ldots, m\}. \end{aligned} \tag{2.6}$$

Inequality constraints can be divided into active and inactive ones, defined by the subsequent definition.

Definition 2.2. *The constraint* $g_j,\ j \in \{m_e + 1, \ldots, m\}$ *is active at* $x \in \mathbb{F}$*, if*

$$g_j(x) = 0 \tag{2.7}$$

holds. Moreover, the active set at $x \in \mathbb{F}$ *is defined by*

$$\mathbb{A}(x) := \{j \in \{1, \ldots, m\} : g_j(x) = 0\}. \tag{2.8}$$

The process of finding a feasible point $x^* \in \mathbb{F}$, which minimizes the objective function $f(x)$ of the nonlinear program (2.1) is called nonlinear programming. We distinguish between local and global minima.

Definition 2.3. $x^* \in \mathbb{F}$ *is a local minimum of NLP* (2.1)*, if there exists a neighborhood* $U(x^*)$ *such that*

$$f(x^*) \leq f(x) \tag{2.9}$$

holds, for all $x \in U(x^*) \cap \mathbb{F}$.

$x^* \in \mathbb{F}$ *is a global minimum of* (2.1)*, if*

$$f(x^*) \leq f(x) \tag{2.10}$$

holds, for all $x \in \mathbb{F}$.

The nonlinear program (2.1) is associated with the Lagrangian function combining the objective and the constraints via the Lagrangian multipliers.

Definition 2.4. *The Lagrangian function of NLP* (2.1) *is given by*

$$L(x, \lambda, \mu) := f(x) - \sum_{j \in \mathbb{J}_=} \mu_j g_j(x) - \sum_{j \in \mathbb{J}_>} \lambda_j g_j(x) \tag{2.11}$$

where $\mu \in \mathbb{R}^{m_e}$ *and* $\lambda \in \mathbb{R}^{m-m_e}$ *with* $\lambda_j \geq 0,\ \forall j \in \mathbb{J}_>$ *are called Lagrangian multipliers. The Lagrangian multipliers are considered as dual variables, while* $x \in \mathbb{R}^n$ *are the primal variables.*

The introduction of Lagrangian multipliers $\mu \in \mathbb{R}^{m_e}$ and $\lambda \in \mathbb{R}^{m-m_e}$ allows the distinction of active constraints according to Definition 2.5.

Definition 2.5. *The active constraint* $g_j(x),\ j \in \mathbb{A}(x)$ *and at* $x \in \mathbb{F}$ *is strongly active, if the corresponding Lagrangian multiplier* λ_j *or* μ_j *respectively, is not equal to zero. The set of strongly active constraints is denoted by* $\mathbb{A}_S(x)$.

We want to introduce necessary and sufficient optimality criteria for nonlinear programming. Especially the necessary first order optimality conditions describing a KKT point given in Theorem 2.1 is of practical importance, since it is frequently used as a stopping criterion by NLP solvers. A regularity condition has to be satisfied to guarantee the existence of a KKT point. One sufficient condition is the linear independence constraint qualification (LICQ).

Definition 2.6. *The linear independence constraint qualification (LICQ) is satisfied at a feasible point* $x \in \mathbb{F}$*, if the gradients of the constraints included in the active set* $\mathbb{A}(x)$ *are linear independent at* x.

First order necessary optimality conditions for NLP (2.1) can be stated as follows.

Theorem 2.1. *Let* $x^* \in \mathbb{F}$ *be a local minimum of* (2.1)*, where* f *and* g *are twice continuously differentiable, and let the LICQ be satisfied at* x^**, see Definition 2.6.*

Then there exist $\mu^* \in \mathbb{R}^{m_e}$ *and* $\lambda^* \in \mathbb{R}^{m-m_e}$ *such that the following Karush-Kuhn-Tucker (KKT) conditions hold*

$$\begin{aligned} \nabla_x L(x^*, \lambda^*, \mu^*) &= 0, && (2.12)\\ g_j(x^*) &= 0,\ j = 1, \ldots, m_e, && (2.13)\\ g_j(x^*) &\geq 0,\ j = m_e + 1, \ldots, m, && (2.14)\\ \lambda_j^* &\geq 0,\ j = m_e + 1, \ldots, m, && (2.15)\\ \lambda_j^* g_j(x^*) &= 0,\ j = m_e + 1, \ldots, m. && (2.16) \end{aligned}$$

If (x^*, λ^*, μ^*) *satisfies the KKT conditions, i.e.,* (2.12) - (2.16)*, it is called KKT-point or stationary point.*

Proof. See e.g., Jarre and Stoer [66]. □

The Lagrangian multipliers (λ^*, μ^*) are unique, if the LICQ condition holds, see Schittkowski and Yuan [96]. Second order sufficient optimality conditions are given by the subsequent theorem, see, e.g., Jarre and Stoer [66].

Theorem 2.2. *Let the following conditions be satisfied for NLP* (2.1)*:*

1. $f(x) \in C^2(\mathbb{R}^n)$ *and* $g(x) = (g_1(x), \ldots, g_m(x))^T \in C^2(\mathbb{R}^n)$.
2. *Let* $(x^*, \lambda^*, \mu^*) \in \mathbb{R}^{n \times m}$ *be a KKT-point of NLP* (2.1).
3. *For all* $s \in L(x^*)$ *with*

$$L(x^*) := \left\{ s \in \mathbb{R}^n : \begin{array}{rcll} \nabla_x g_j(x^*)^T s &=& 0, & j \in \mathbb{A}_S(x^*),\\ \nabla_x g_j(x^*)^T s &\geq& 0, & j \in (\mathbb{A}(x^*) \backslash \mathbb{A}_S(x^*)) \cap \mathbb{J}_> \end{array} \right\} \quad (2.17)$$

and $s \neq 0$*,*

$$s^T \nabla_x^2 L(x^*, \lambda^*, \mu^*) s > 0 \quad (2.18)$$

holds, where $\mathbb{A}_S(x^*)$ *is the set of strongly active constraints at* $x^* \in \mathbb{F}$*, see Definition 2.5.*

Then x^* *is a strict local minimum of NLP* (2.1)*, i.e.,* $\exists\ \varepsilon_1 > 0,\ \varepsilon_2 > 0$*, such that* $\forall\ x \in \{x \in \mathbb{F} : \|x - x^*\| < \varepsilon_2\}$

$$f(x) \geq f(x^*) + \varepsilon_1 \|x - x^*\|^2 \quad (2.19)$$

holds.

Proof. See e.g., Jarre and Stoer [66]. □

Many nonlinear programming algorithms incorporate second order information. In the remainder, we denote the Hessian of the Lagrangian function with respect to x by

$$H(x) \;:=\; \nabla_x^2 L(x, \lambda, \mu) \in \mathbb{R}^{n \times n}. \tag{2.20}$$

In many applications, the evaluation of $H(x)$ is too expensive. Furthermore, the Hessian matrix might be indefinite. Approximating $H(x)$ via appropriate updating schemes, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update given by Definition 2.7, yields a positive definite approximation, which can be computed efficiently, see e.g., Schittkowski and Yuan [96] or Jarre and Stoer [66]. We denote the Hessian or its approximation in iteration k by $B^k \in \mathbb{R}^{n \times n}$.

Definition 2.7. *The BFGS Quasi-Newton Update in iteration k is given by*

$$B^{k+1} \;:=\; B^k + \frac{d_L^k (d_L^k)^T}{(d^k)^T d_L^k} - \frac{B^k d^k (d^k)^T B^k}{(d^k)^T B^k d^k} \tag{2.21}$$

with

$$d_L^k \;:=\; \nabla_x L(x^{k+1}, \lambda^k, \mu^k) - \nabla_x L(x^k, \lambda^k, \mu^k) \in \mathbb{R}^n, \tag{2.22}$$

$$d^k \;:=\; x^{k+1} - x^k \in \mathbb{R}^n. \tag{2.23}$$

d^k is also called search direction. The matrix B^0 is commonly initialized by the identity matrix of appropriate dimension.

In general, NLP (2.1) is solved iteratively by a sequence of subproblems, which are efficiently solvable. At each iterate $x^k \in X$ a subproblem is formulated and the next iterate is determined by the solution of the current subproblem. The sequence of iterates $\{x^k\}$ is supposed to converge towards a stationary point, if certain requirements are satisfied.

In nonlinear programming, local and global convergence properties are distinguished. The local convergence behavior of an algorithm describes the rate of convergence in the neighborhood of a stationary point $x^* \in \mathbb{F}$. We distinguish three different convergence rates.

Definition 2.8. *An iteration sequence $\{x^k\}$ converges with linear convergence rate towards a stationary point $x^* \in \mathbb{F}$, if there exists a $c_1 \in]0, 1[$ with*

$$\|x^{k+1} - x^*\| \;\leq\; c_1 \|x^k - x^*\|, \tag{2.24}$$

for all $k > \bar{k}$ for some fixed $\bar{k}$.

An iteration sequence $\{x^k\}$ converges with superlinear convergence rate towards a stationary point $x^* \in \mathbb{F}$, if there exists a sequence $\{c^k\} \subset \mathbb{R}_+$, satisfying $\lim_{k \to \infty} c^k = 0$, and

$$\|x^{k+1} - x^*\| \;\leq\; c^k \|x^k - x^*\|, \tag{2.25}$$

holds for all $k > \bar{k}$ *for some fixed* $\bar{k}$.

An iteration sequence $\{x^k\}$ *converges with quadratic convergence rate towards a stationary point* $x^* \in \mathbb{F}$*, if there exists a finite constant* $c_2 > 0$ *with*

$$\|x^{k+1} - x^*\| \leq c_2 \|x^k - x^*\|^2, \tag{2.26}$$

for all $k > \bar{k}$ *for some fixed* $\bar{k}$.

The property, that an algorithm converges towards a stationary point x^* independently of the starting point $x^0 \in X$ is called global convergence. Note, that a globally convergent NLP solver, does not guarantee to find a global minimum according to Definition 2.3, instead it finds a stationary point from any starting point $x^0 \in X$.

Global convergence properties are ensured by so-called globalization techniques. In the literature, different approaches can be found, such as

1. trust region methods, see Vardi [103], Byrd, Schnabel and Shultz [35] and Omojokun [84],
2. filter methods, see Fletcher and Leyffer [51],
3. and line search methods, see Armijo [6].

The well-known and widely used sequential quadratic programming (SQP) methods, converge with a superlinear convergence rate towards a KKT point, if appropriate requirements are satisfied, see Schittkowski and Yuan [96]. An essential assumption is, that the search step determined by the SQP subproblem, is accepted without adjustments in a neighborhood of a KKT point. This might be prevented by applying globalization techniques, i.e., they might force a modification of the search direction even in a neighborhood of the KKT point. This phenomenon is known as the Maratos effect, see Maratos [78].

Many techniques have been proposed to prevent the Maratos effect, e.g.,

1. watch-dog techniques, see Chamberlain et al. [39],
2. second order correction techniques, see Fletcher [49], Mayne and Polak [79] and Fukushima [56],
3. usage of smooth exact penalty functions or the augmented Lagrange function as merit function, see Schittkowski [93], Powell and Yuan [88],[89] and Ulbrich [101],
4. non-monotone techniques, see Ulbrich and Ulbrich [102] and Gould and Toint [60].

In our implementation of MIQPSOA, which is proposed in Chapter 3, second order correction steps are applied to avoid the Maratos effect. A second order correction step tries to correct constraint violations, that are caused by the SQP search step by considering second order information. The second order correction step is obtained by solving a second subproblem in the corresponding iteration, see e.g., Yuan [112] for further details.

2.2 Sequential Quadratic Programming and the Trust Region Method of Yuan

In this section, we review a trust region algorithm yielding a stationary point for the continuous nonlinear program (2.1). To simplify readability we assume that the box constraints are included in the nonlinear constraints $g(x)$, see (2.5). It was proposed by Yuan [112] and is closely related to the well-known sequential quadratic programming methods. The most common nonlinear programming algorithms are interior point methods, e.g., Wächter and Biegler [105] or sequential quadratic programming (SQP) methods, e.g., Schittkowski and Yuan [96]. These nonlinear programming algorithms converge globally towards a stationary point. A global minimum can only be guaranteed for convex problems, e.g., the continuous relaxation of the convex MINLP (1.6).

Nonlinear programming problems naturally arise during the solution process of convex mixed-integer nonlinear optimization problems (1.6), e.g., as continuous relaxation or by fixing the integer variables $y \in \mathbb{N}^{n_i}$. We focus on the trust region method proposed by Yuan [112], since it is the base for a novel MINLP solution method presented in Section 3.1.

Every SQP method is based on solving a series of continuous quadratic (QP) subproblems. In iteration k a quadratic subproblem is constructed by linearizing the constraints at the current iterate x^k. The objective function is a quadratic approximation of the Lagrangian function $L(x, \lambda, \mu)$ of NLP (2.1), see Definition 2.4 and Definition 2.7 for a suitable choice of the matrix B^k forming the quadratic term in the objective function. The subproblem in iteration k is given by

$$
\begin{aligned}
& d \in \mathbb{R}^n : \\
& \qquad \begin{aligned}
\min \quad & \nabla_x f(x^k)^T d + \tfrac{1}{2} d^T B^k d \\
\text{s.t.} \quad & g_j(x^k) + \nabla_x g_j(x^k)^T d \;=\; 0, \quad j = 1, \ldots, m_e, \\
& g_j(x^k) + \nabla_x g_j(x^k)^T d \;\geq\; 0, \quad j = m_e + 1, \ldots, m.
\end{aligned}
\end{aligned}
\tag{2.27}
$$

The solution of QP (2.27) is denoted by d^k. The next iterate x^{k+1} is obtained either directly, i.e., $x^{k+1} = x^k + d^k$, if a trust region constraint $d \leq \|\Delta^k\|$ is included, where Δ^k denotes the trust region radius. Or it is obtained by a line search, i.e., $x^{k+1} = x^k + \alpha^k d^k$, with $\alpha^k \in (0, 1]$ reducing the step-length α^k until a sufficient descent with respect to an appropriate merit-function is obtained.

The subproblems of the trust region method of Yuan [112] approximate the L_∞-penalty function introduced in Definition 2.9 instead.

Definition 2.9. *The L_∞-penalty function $P_\sigma(x)$ associated with the penalty parameter $\sigma \in \mathbb{R}_+$ and $x \in \mathbb{R}^n$ is given by*

$$
P_\sigma(x) \;=\; f(x) + \sigma \|g(x)^-\|_\infty, \tag{2.28}
$$

with

$$g(x)^- := (g_1(x)^-, \ldots, g_m(x)^-)^T \tag{2.29}$$

$$g_j(x)^- := \begin{cases} g_j(x), & j = 1, \ldots, m_e \\ \min\{g_j(x), 0\}, & j = m_e + 1, \ldots, m. \end{cases} \tag{2.30}$$

In each iteration k the trust region method constructs a model of the original problem based on the current iterate x^k. Typically, the accuracy and therefore the quality of the model decreases with an increasing distance from x^k. To control the quality of the model, the maximal distance from the current iterate is restricted by the trust region radius Δ^k.

In iteration k Yuan's trust region algorithm approximates the L_∞-penalty function leading to the following problem:

$$\begin{aligned} & d \in \mathbb{R}^n : \\ & \qquad \min \quad \Phi^k(d) \\ & \qquad \text{s.t.} \quad \|d\|_\infty \leq \Delta^k, \end{aligned} \tag{2.31}$$

where the objective function is given by

$$\begin{aligned} \Phi^k(d) := & \quad \nabla_x f(x^k)^T d + \frac{1}{2} d^T B^k d \\ & + \sigma^k \|(g(x^k) + [\nabla_x g(x^k)]^T d)^-\|_\infty. \end{aligned} \tag{2.32}$$

Note, that problem (2.31) is equivalent to a quadratic program, see Yuan [112]. Therefore, it can be solved efficiently by any QP solver, e.g., QL of Schittkowski [94]. The optimal solution of subproblem (2.31) is denoted by d^k. It provides a search direction to determine the next iterate, i.e.,

$$x^{k+1} = x^k + d^k. \tag{2.33}$$

Problem (2.31) predicts a decrease in the L_∞-penalty function. We define

$$\mathrm{Pred}^k := \Phi^k(0) - \Phi^k(d^k). \tag{2.34}$$

If the subsequent stopping criterion

$$d^k = 0 \tag{2.35}$$

and certain assumptions hold, the current iterate x^k is one out of three stationary points, that are defined below. One of these stationary points corresponds to a KKT-point of NLP (2.1), see Yuan [112].

To reduce the constraint violation a penalty parameter is introduced, that penalizes the violation of constraints. If feasibility is not sufficiently improved, the penalty

parameter σ^k has to be increased. The increase of the penalty parameter is controlled by a parameter $\delta^k \in \mathbb{R}$ with $\delta^k > 0$.

Comparing the predicted reduction given by (2.34), with the realized decrease of the L_∞-penalty function, the quality of model associated with problem (2.31) can be judged. The corresponding measure is denoted by

$$r^k := \frac{P_{\sigma^k}(x^k) - P_{\sigma^k}(x^k + d^k)}{Pred^k}. \tag{2.36}$$

If the model corresponding to problem (2.31) is of poor quality, e.g. $r^k \leq 0$, the trust region radius has to be decreased. Otherwise, the trust region radius can be extended. Subsuming all components we can specify the trust region method of Yuan [112].

Algorithm 2.1. (Trust Region Method of Yuan)

1. *Given* $x^0 \in R^n$, $\Delta^0 > 0$, $B^0 \in \mathbb{R}^{n\times n}$ *symmetric and positive definite,* $\delta^0 > 0$, $\sigma^0 > 0$ *and* $k := 0$.

   Evaluate the functions $f(x^0)$ *and* $g(x^0)$ *and determine gradients* $\nabla_x f(x^0)$ *and* $\nabla_x g(x^0)$.

2. *Solve subproblem* (2.31) *determining* d^k.

   If *stopping criterion* (2.35) *is satisfied,* **then STOP**.

   Else *evaluate* $f(x^k + d^k)$, $g(x^k + d^k)$ *and* $P_{\sigma^k}(x^k + d^k)$.
   Evaluate the quality of the model with respect to the L_∞*-penalty function by* r^k (2.36).

3. *Check for a descent with respect to the* L_∞*-penalty function.*

   If $r^k > 0$, **then** *update iterate, i.e., set* $x^{k+1} := x^k + d^k$.

   Else *set* $x^{k+1} := x^k$, $B^{k+1} := B^k$, $\Delta^{k+1} := \frac{1}{4}\|d^k\|_\infty$, $k := k + 1$.
   GOTO *Step 2.*

4. *Adapt trust region radius:*

$$\Delta^{k+1} := \begin{cases} \max\{\Delta^k, 4\|d^k\|_\infty\}, & \text{if } r^k > 0.9, \\ \Delta^k, & \text{if } 0.9 \geq r^k \geq 0.1, \\ \min\{\frac{1}{4}\Delta^k, \frac{1}{2}\|d^k\|_\infty\}, & \text{if } r^k < 0.1. \end{cases} \tag{2.37}$$

   Choose B^{k+1}, *such that* B^{k+1} *is any symmetric, positive definite matrix.*

   Penalty Update:

If

$$\Phi^k(0) - \Phi^k(d^k) \leq \sigma^k \delta^k \min\left\{\Delta^k, \|g(x^k)^-\|_\infty\right\}, \tag{2.38}$$

then *set*

$$\sigma^{k+1} := 2\sigma^k, \tag{2.39}$$

$$\delta^{k+1} := \frac{1}{4}\delta^k. \tag{2.40}$$

Else *set*

$$\sigma^{k+1} := \sigma^k, \tag{2.41}$$

$$\delta^{k+1} := \delta^k. \tag{2.42}$$

5. *Compute* $\nabla_x f(x^{k+1})$ *and* $[\nabla_x g(x^{k+1})]$, *set* $k := k+1$ *and* **GOTO** *Step 2.*

The trust region method of Yuan converges towards a stationary point, if Assumption 2.1 is satisfied, see Yuan [112] and Jarre and Stoer [66].

Assumption 2.1. *1.* $f(x)$ *and* $g_j(x),\ j = 1, \ldots, m$ *are continuously differentiable* $\forall x \in \mathbb{R}^n$.

2. The sequences $\{x^k\}$ *and* $\{B^k\}$ *are bounded* $\forall\ k$.

Moreover, we review the theoretical properties of Algorithm 2.1, according to Yuan [112]. The limit of $\|g(x^k)^-\|_\infty$ exists, if the sequence of penalty parameter $\{\sigma^k\}$ goes to infinity.

Lemma 2.1. *If Assumption 2.1 holds and* $\sigma^k \to \infty$, *then* $\lim_{k\to\infty} \|g(x^k)^-\|_\infty$ *exists.*

Proof. See Yuan [112]. □

Algorithm 2.1 converges towards one of three different stationary points, see Yuan [112]. These are either a KKT-point for problem (2.1) as specified in Theorem 2.1. Or it is a infeasible stationary point or a singular stationary point.

Yuan [112] defines an infeasible stationary point as follows.

Definition 2.10. *Let* $\mathbb{F}$ *be the feasible region of NLP* (2.1). $x^* \in \mathbb{R}^n \backslash \mathbb{F}$ *is called an infeasible stationary point, if*

1. $\|g(x^*)^-\|_\infty > 0$
2. $\min_{d\in\mathbb{R}^n} \|(g(x^*) + [\nabla_x g(x^*)]^T d)^-\|_\infty = \|g(x^*)^-\|_\infty$

holds.

An infeasible stationary point is a minimizer of the infinity norm of the linearized constraints, see Yuan [112]. In contrast to an infeasible stationary point, a singular stationary point is feasible. It is defined as follows.

Definition 2.11. *Let $\mathbb{F}$ be the feasible region of NLP (2.1). $x^* \in \mathbb{F}$ is called a singular stationary point, if the following conditions hold.*

1. $\|g(x^*)^-\|_\infty = 0$.
2. *There exists a sequence $\{\hat{x}^k\}$ converging towards x^* such that $\|g(\hat{x}^k)^-\|_\infty > 0$ and*

$$\lim_{k\to\infty} \min_{\|d\|_\infty \le \|g(\hat{x}^k)^-\|_\infty} \frac{\|(g(\hat{x}^k) + [\nabla_x\, g(\hat{x}^k)]^T d)^-\|_\infty}{\|g(\hat{x}^k)^-\|_\infty} = 1. \tag{2.43}$$

In addition a stationary point is defined as follows.

Definition 2.12. *Let $\mathbb{F}$ be the feasible region of NLP (2.1). $x^* \in \mathbb{F}$ is called a stationary point, if the following conditions hold.*

1. $\|g(x^*)^-\|_\infty = 0$.
2. *If*

$$\nabla_x g_j(x^*)^T s \;\ge\; 0,\; j \in \mathbb{J}_> \tag{2.44}$$

$$\nabla_x g_j(x^*)^T s \;=\; 0,\; j \in \mathbb{J}_= \tag{2.45}$$

hold for $s \in \mathbb{R}^n$, then

$$\nabla_x f(x^*)^T s \;\ge\; 0 \tag{2.46}$$

is satisfied

Note, that a stationary point according to Definition 2.12 corresponds to a KKT-point of NLP (2.1), see Jarre and Stoer [66].

The following theorem shows, that the trust region method of Yuan converges towards a KKT point, if the sequence of penalty parameters $\{\sigma^k\}$ is bounded.

Theorem 2.3. *If Assumption 2.1 holds and the sequence of penalty parameters $\{\sigma^k\}$ is bounded, one member of the sequence of iterates $\{x^k\}$ is a stationary point specified in Definition 2.12 or the sequence $\{x^k\}$ generated by Algorithm 2.1 possesses an accumulation point that is a stationary point.*

Proof. *See Yuan [112].*

If the sequence of penalty parameters $\{\sigma^k\}$ is unbounded and the constraints are violated in the limit, Yuan's trust region method converges towards an infeasible stationary point given by Definition 2.10.

Lemma 2.2. *If Assumption 2.1 holds,* $\lim_{k\to\infty} \sigma^k = \infty$ *and* $\lim_{k\to\infty} \|g(x^k)^-\|_\infty > 0$*, then the sequence* $\{x^k\}$ *possesses an infeasible stationary point of NLP* (2.1) *as an accumulation point.*

Proof. *See Yuan [112].*

If the sequence of penalty parameters $\{\sigma^k\}$ is unbounded but the constraint violation goes to zero, Yuan's trust region method converges towards a singular stationary point as specified in Definition 2.11.

Lemma 2.3. *If Assumption 2.1 holds,* $\lim_{k\to\infty} \sigma^k = \infty$ *and* $\lim_{k\to\infty} \|g(x^k)^-\|_\infty = 0$*, then the sequence* $\{x^k\}$ *generated by Algorithm 2.1 possesses a singular stationary point of NLP* (2.1) *as an accumulation point.*

Proof. See Yuan [112]. □

2.3 Review on Solution Techniques for Convex MINLPs

There exists a variety of different algorithms to solve convex mixed-integer nonlinear optimization problems given by problem (1.6), see, e.g., Floudas [52], Grossmann and Kravanja [61] or Bonami, Kilinc and Linderoth [28] for review papers. This thesis focuses on the development of solution methods for convex MINLP problems based on the successive solution of mixed-integer quadratic subproblems. Our new algorithm called MIQP-Supported Outer Approximation (MIQPSOA) proposed in Chapter 3 as well as the solution methods, that are reviewed in this chapter rely on gradient information.

In general, for solving any mixed-integer optimization problem the existence of lower and upper bounds is of major importance. These bounds refer to the value of the objective function of the mixed-integer program. For a lower bound $\underline{f}$, $f(x^*, y^*) \geq \underline{f}$ holds at the optimal solution (x^*, y^*), while $f(x^*, y^*) \leq \overline{f}$ holds for an upper bound $\overline{f}$.

The existing solution methods can be classified into branch-and-bound algorithms, methods based on the successive solution of mixed-integer linear relaxations and advanced methods integrating continuous nonlinear and mixed-integer techniques.

Branch-and-bound methods are commonly used to solve non-convex MINLPs (1.1) by exploiting analytical problem structures within a global solver, such as BARON, see Sahinidis [92]. NLP-based branch-and-bound methods do not rely on any analytical knowledge of the problem and are often applied to solve the convex MINLP (1.6).

Although we concentrate on the development of competitive solution approaches we briefly review the basic concepts of general branch-and-bound algorithms and NLP-based branch-and-bound in Section 2.4. Furthermore, a basic implementation is used as benchmark for our numerical tests in Chapter 6.

Another class of algorithms relies on the successive solution of mixed-integer linear programming (MILP) relaxations of the convex MINLP (1.6). Among these methods there is the extended cutting plane method proposed by Westerlund and Pettersson [108], see also Westerlund and Pörn [109] and Section 2.7 for details. Linear outer approximation and the generalized Benders' decomposition proposed by Geoffrion [57] are other frequently used algorithms. The basic concept of linear outer approximation methods is introduced by Duran and Grossmann [43] and is extended by Fletcher and Leyffer [50]. We present the basic theory of the linear outer approximation approach of Fletcher and Leyffer [50] in Section 2.5. Generalized Benders' decomposition is a similar approach, that is briefly described in Section 2.6.

The last class of solvers for convex MINLPs contains advanced, state-of-the-art solution methods. Originally, continuous nonlinear and mixed-integer optimization is decoupled, since efficient black-box solvers for both problems are available. The advanced methods improve the performance by integrating both techniques. This class is constituted by LP/NLP-based branch-and-bound proposed by Quesada and Grossman [90], see Section 2.8 and nonlinear branch-and-bound with early branching according to Leyffer [76], which is reviewed in Section 2.9.

Convex MINLP problems are hard to solve and each available algorithm, such as NLP-based branch-and-bound, linear outer approximation, generalized Benders' decomposition, LP/NLP-based branch-and-bound and the extended cutting plane method, is based on the successive solution of complex subproblems. Since the problem class of mixed-integer nonlinear optimization problems is the combination of mixed-integer linear programming and continuous nonlinear programming, these subproblems are either traditionally MILPs or NLPs.

Based on Grossmann [62] we are going to review the interrelation of these solution methods by analyzing their subproblems. To ease the readability we omit linear equality constraints, i.e., $\mathbb{J}_{=} = \emptyset$ and $\mathbb{J} = \mathbb{J}_{>}$.

Essentially, three different continuous nonlinear subproblems might arise during the solution of the convex MINLP problem (1.6). One subproblem, which is associated with NLP-based branch-and-bound methods, is the continuous relaxation denoted by NLP_r

$$\begin{aligned} x \in X,\ y \in Y_{\mathbb{R}}: \\ \min \quad & f(x,y) \\ \text{s.t.} \quad & g_j(x,y) \ \geq \ 0, \quad j \in \mathbb{J}, \end{aligned} \tag{2.47}$$

where $Y_{\mathbb{R}}$ is defined by (1.8). The optimal objective value of the continuous relaxation provides a lower bound on the optimal solution of MINLP (1.6). NLP-based branch-and-bound methods successively solve problem (2.47). In each iteration the

box-constraints of the integer variables are refined, i.e., replaced by more stringent lower or upper ones, which are given by

$$y_i \;\geq\; (y_l^k)_i \;>\; (y_l)_i, \quad i \in I_L^k \tag{2.48}$$

or

$$y_i \;\leq\; (y_u^k)_i \;<\; (y_u)_i, \quad i \in I_U^k. \tag{2.49}$$

I_L^k, I_U^k denote the corresponding index subsets of the integer variables y_i, $i \in I$, with $(y_l)_i < (y_l^k)_i$ or $(y_u^k)_i < (y_u)_i$ respectively, in iteration k, i.e.,

$$I_L^k \;:=\; \{i \in I : \exists (y_l^k)_i : y_i \geq (y_l^k)_i > (y_l)_i \text{ in iteration } k\}, \tag{2.50}$$

or

$$I_U^k \;:=\; \{i \in I : \exists (y_u^k)_i : y_i \leq (y_u^k)_i < (y_u)_i \text{ in iteration } k\}. \tag{2.51}$$

The remaining two continuous nonlinear subproblems arise in linear outer approximation, generalized Benders' decomposition and LP/NLP-based branch-and-bound. They are both derived from the original mixed-integer nonlinear program (1.6) by fixing the integer variables to a certain integer value $y^k \in Y$. k denotes the k-th integer value, that is considered. Fixing the integer variables $y \equiv y^k$ yields the continuous problem NLP(y^k) given by

$$\begin{aligned} &x \in X: \\ &\qquad \min \quad f(x, y^k) \\ &\qquad \text{s.t.} \quad g_j(x, y^k) \;\geq\; 0, \quad j \in \mathbb{J}. \end{aligned} \tag{2.52}$$

Its solution $\bar{x}_{y^k} \in X$ provides an upper bound for the mixed-integer nonlinear program (1.6). If problem NLP(y^k) does not possess a feasible solution, i.e.,

$$\nexists\, x \in X: \; g_j(x, y^k) \geq 0, \; \forall j \in \mathbb{J}, \tag{2.53}$$

for some integer value $y^k \in Y$, subproblem (2.52) has to be replaced by the so-called feasibility subproblem. The goal is to find a value of $x \in X$, such that the constraints are least violated in some norm for fixed $y^k \in Y$. Choosing the ∞-norm, the nonlinear feasibility problem F(y^k) is given by

$$\begin{aligned} &x \in X, \; \eta_F \in \mathbb{R}_+ : \\ &\qquad \min \quad \eta_F \\ &\qquad \text{s.t.} \quad g_j(x, y^k) \;\geq\; -\eta_F, \quad j \in \mathbb{J}. \end{aligned} \tag{2.54}$$

and we denote its solution by $(\bar{x}_{y^k}, \bar{\eta}_F) \in X \times \mathbb{R}_+$. $\eta_F \in \mathbb{R}_+$ is an additional variable measuring the maximal constraint violation for $g_j(x, y)$, $j \in \mathbb{J}$. Note; that problem (2.54) minimizes the maximal constraint violation, i.e., the $\|.\|_\infty$-norm of the constraint violation. There exist many variants of feasibility problem (2.54), e.g., using different

norms measuring the constraint violation. In the sequel, all variants are denoted by $F(y^k)$, since a distinction is not necessary.

Some algorithms for solving convex MINLPs rely on the successive solution of mixed-integer linear programming problems, which are obtained by relaxing the nonlinear functions of the convex MINLP (1.6). These relaxations are denoted by MILP_r^k and often called master problem. The master problem is successively refined by additional linearizations obtained in previous iterations $i \leq k$ at some points $(x^i, y^i) \in X \times Y_{\mathbb{R}}$. In iteration k, the master problem is given by

$$
\begin{aligned}
& x \in X,\ y \in Y,\ \eta \in \mathbb{R}: && (2.55)\\
& \min \quad \eta \\
& \text{s.t.} \quad f(x^i, y^i) + \nabla_{x,y} f(x^i, y^i)^T \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} \leq \eta, \quad \forall (x^i, y^i) \in \mathbb{L}^k, \\
& \qquad g_j(x^i, y^i) + \nabla_{x,y} g_j(x^i, y^i)^T \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} \geq 0, \quad \forall (x^i, y^i) \in \mathbb{L}^k, \quad j \in \mathbb{J}^i,
\end{aligned}
$$

with $\mathbb{J}^i \subseteq \mathbb{J},\ \forall\, i = 1, \ldots, k$, and $\mathbb{L}^k := \{(x^i, y^i) \in X \times Y_{\mathbb{R}}, i = 1, \ldots, k\}$. The set $\mathbb{J}^i$ in iteration i is a subset of $\mathbb{J}$, which can vary in each iteration $i = 1, \ldots, k$. Furthermore, its definition depends on the corresponding algorithm. $(x^i, y^i) \in X \times Y_{\mathbb{R}},\ i = 1, \ldots, k$ are linearization points obtained in iteration $i \leq k$. For a convex MINLP (1.6) the corresponding MILP_r (2.55) is defined such that the constraints

$$
f(x^i, y^i) + \nabla_{x,y} f(x^i, y^i)^T \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} \leq \eta, \quad \forall (x^i, y^i) \in \mathbb{L}^k \qquad (2.56)
$$

underestimate the objective function $f(x, y)$. Figure 2.1 illustrates the situation, the objective function f is underestimated by two linearizations (2.56) denoted by $\mathtt{lin}_1(f)$ and $\mathtt{lin}_2(f)$. Since the constraints

$$
g_j(x^i, y^i) + \nabla_{x,y} g_j(x^i, y^i)^T \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} \geq 0, \quad j \in \mathbb{J}^k_{>}, \quad \forall (x^i, y^i) \in \mathbb{L}^k \qquad (2.57)
$$

overestimate the feasible region, the master problem (2.55) is a relaxation of the convex MINLP (1.6), for any linearization point $(x^i, y^i) \in \mathbb{L}^k$. Figure 2.2 shows, that the feasible region of a convex MINLP described by $g_1(x, y)$, $g_2(x, y)$ and $g_3(x, y)$ is overestimated by any linearizations $\mathtt{lin}_1$, $\mathtt{lin}_2$ of any of the constraints $g_1(x, y)$, $g_2(x, y)$, $g_3(x, y)$ obtained from (2.57).

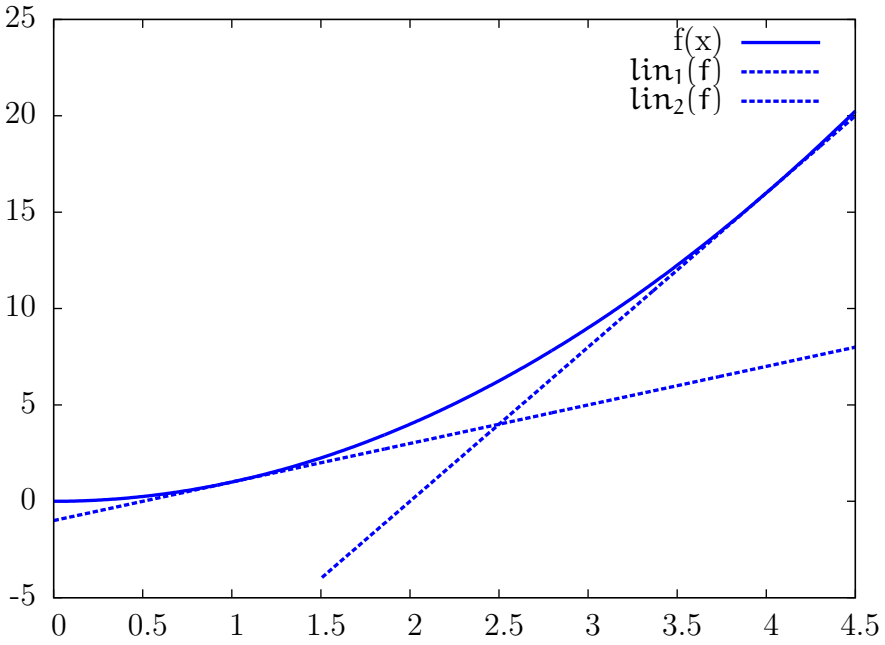

Fig. 2.1: Underestimating the Objective Function

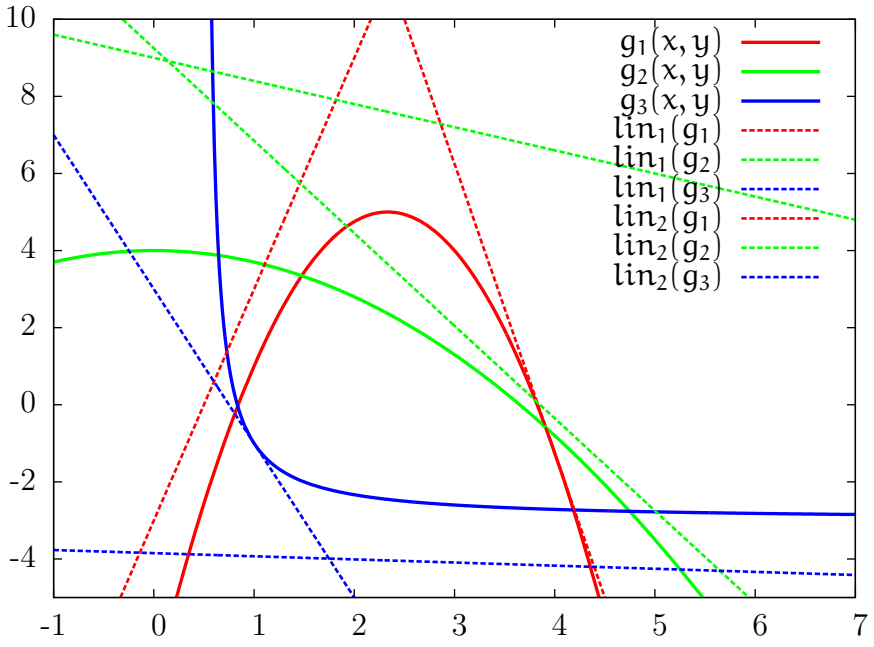

Fig. 2.2: Overestimating the Feasible Region

As a consequence, the value of η^*, which is part of the optimal solution of MILP (2.55), provides a lower bound on the global solution of MINLP (1.6). The linearization points

$(x^i, y^i) \in \mathbb{L}^k$ often correspond to a solution of one of the continuous nonlinear subproblems $\mathrm{NLP}(y^i)$ or $\mathrm{F}(y^i)$. More details concerning the subproblems $\mathrm{NLP}(y^k)$, $\mathrm{F}(y^k)$ and MILP_r^k in the context of linear outer approximation and generalized Benders' decomposition are given in Section 2.5 and Section 2.6, respectively.

2.4 NLP-based Branch-and-Bound

Originally, branch-and-bound methods were developed for solving mixed-integer linear optimization problems, see Dakin [42]. Since branch-and-bound is a general concept, it can easily be applied to solve convex mixed-integer nonlinear programs as shown, e.g., by Gupta and Ravindran [64] or Borchers and Mitchell [29]. An efficient state-of-the-art implementation is provided by Bonami et al. [27].

First, we describe the general concept of branch-and-bound methods, according to Floudas [52]. Moreover, we discuss NLP-based branch-and-bound methods, that are often used to solve the convex MINLP problem (1.6). The concept of branch-and-bound is generally valid and can be applied for any optimization problem P. Usually P is a mixed-integer linear program. But also global optimization methods for non-convex, continuous, nonlinear programs are based on the branch-and-bound concept in combination with convex under- and over-estimators, see Sahinidis [92] for a state-of-the-art implementation.

Branch-and-bound is generally based on partitioning, relaxation and fathoming. During the partition step of a branch-and-bound method, the original problem is iteratively subdivided into a finite number of disjoint subproblems. Furthermore, the solution process is based on a series of relaxations of the original problem, that are easier to solve. The concept of fathoming is very important for the efficiency of a branch-and-bound method. Fathoming denotes the fact, that certain subproblems need not be considered, if they cannot contain an optimal solution. The main goal of a branch-and-bound method is to perform an enumeration of the solutions of an optimization problem P, without explicitly examining all possible alternatives, e.g., all possible values for every integral variable included in P.

After the brief introduction, we present the general branch-and-bound concept in detail. First we define a partition of P.

Definition 2.13. *A partition of an optimization problem P is a set of υ subproblems $P_1, \ldots, P_\upsilon$, such that the following conditions hold:*

1. *A feasible solution of any of the subproblems $P_1, \ldots, P_\upsilon$ is a feasible solution of P.*

2. *Every feasible solution of P is a feasible solution of exactly one of the subproblems $P_1, \ldots, P_\upsilon$.*

Definition 2.13 ensures that the feasible solutions of the subproblems $P_1, \ldots, P_\upsilon$ form a partition of the feasible solutions of the original problem P. Different possibilities exist for generating a partition of problem P into subproblems $P_1, \ldots, P_\upsilon$. For mixed-integer optimization problems, the most common partitioning technique is branching on an integer variable $y_i \in Y$ obtaining the additional constraint $y_i \leq \psi$ in subproblem P_1 and $y_i \geq \psi + 1$ in subproblem P_2 with $\psi \in \mathbb{N}$, respectively. Moreover, we define a relaxation of Problem P by Definition 2.14.

Definition 2.14. *Let problem* RP *and problem* P *have the same objective function. Then problem* RP *is a relaxation of problem* P*, if the set of feasible solutions of* P *is a subset of the set of feasible solutions of* RP*.*

Note, that usually more than one relaxation exists for problem P. The following properties are valid for any optimization problem P and any corresponding relaxation RP, see Floudas [52].

Corollary 2.1. *Let* P *be an optimization problem and let* RP *be any relaxation of* P*. Then the following properties hold.*

1. *If* RP *does not possess a feasible solution, no feasible solution of* P *exists.*
2. *The optimal solution of the relaxation* RP *provides a lower bound on the optimal solution of* P*.*
3. *If the optimal solution of the relaxation* RP *is feasible for* P*, then it is an optimal solution of the original problem* P*.*

There are several possibilities to obtain relaxations RP for problem P. They might be generated by omitting some constraints or by relaxing some/all constraints. An example is problem (2.55), where a selection of the nonlinear constraints of MINLP (1.6) is replaced by linear ones, such that the feasible region is overestimated. For mixed-integer optimization problems the most common way to obtain a relaxation, is to relax integer variables, i.e., replacing integer variables by continuous ones. The relaxation should be chosen such that the relaxed problem RP is significantly easier to solve than the original problem P. Furthermore, the relaxation is stronger, i.e., more favorable, if the quality of the corresponding lower bound on the optimal solution of P is high. The quality of a lower bound is given by the so-called gap, which is the difference between the optimal objective value of the relaxation RP and the optimal objective value of P. Relaxations that are easy to solve usually provide lower bounds of poor quality.

By iteratively applying the principles of partitioning and relaxation, all possible solutions of optimization problem P are enumerated. This enumeration can be graphically represented by a tree, which is also called branch-and-bound tree, see Figure 2.3. Each node of the tree corresponds to a relaxation of one so-called candidate subproblems (CS).

Definition 2.15. *An unexplored problem* P_i, $i \in \{1, \ldots, v\}$*, that is obtained from problem* P *by partitioning according to Definition 2.13, is called candidate subproblem (CS).*

During the enumeration by branch-and-bound, nodes can be fathomed according to Lemma 2.4, i.e., the corresponding subtrees are cut off without further exploration. A node can always be fathomed, if the corresponding candidate subproblem (CS) possesses no optimal solution for the original problem P, see e.g., Floudas [52].

Lemma 2.4. *A candidate subproblem* CS *can be fathomed, if one of the following conditions hold:*

1. CS *does not contain a feasible solution of* P *possessing a lower objective function value, than the best feasible solution found so far.*
2. *An optimal solution of* CS *is determined.*

Proof. See Floudas [52]. □

There are three general fathoming criteria based on the relaxation of CS, denoted by RCS, see again e.g., Floudas [52].

Corollary 2.2. *A candidate subproblem can be fathomed, if one of the following conditions holds.*

1. *The relaxation* RCS *of* CS *possesses no feasible solution, i.e.,* CS *has no feasible solution.*
2. *The optimal solution of* RCS *is greater or equal than the best known feasible solution of* P*.*

   Note, that this fathoming criterion can be applied more often if the relaxation is tight, i.e., the gap between RCS *and* CS *is small. Obtaining a feasible solution of* P *possessing an objective value as close as possible to the one of the optimal solution of* P *is also of major importance for this fathoming rule.*
3. *The optimal solution of* RCS *is feasible for the candidate problem* CS*, i.e., it is the optimal solution of* CS*.*

   Note, that any feasible solution of CS *is feasible for the original problem* P *and hence possibly the incumbent, i.e., the best feasible solution for* P *known so far, can be updated.*

Most often the relaxations RCS are solved to optimality before appropriate fathoming rules are applied. Nevertheless, one could carry out fathoming before the optimal solution of RCS is obtained by deriving sufficient conditions for the fathoming criteria, e.g., using good suboptimal solutions of the dual problem of P. In addition, post

optimality tests can be applied to improve the quality of the lower bounds obtained by the relaxation RCS.

Figure 2.3 illustrates the concept of the general branch-and-bound method by representing the candidate subproblems CS via a binary search tree. At the root node of the binary search tree we solve a relaxation of the original problem. If the solution of the relaxation is not feasible for the original problem P we partition the root node into two or more candidate subproblems. The candidate subproblems are inserted into a list, containing all existing candidate subproblems. Then one of these problems is selected and the corresponding relaxation is solved. If the relaxation solves the candidate subproblem, the next problem of the candidate list is selected. Otherwise, this subproblem is partitioned and the corresponding problems are included in the candidate list. This process is continued until the candidate list is exhausted. The optimal solution is then given by the current incumbent.

If a branch-and-bound approach generates a finite number of nodes, i.e., the number of candidate subproblems CS is finite, the method guarantees to determine the optimal solution of the original problem P. The fathoming rules allow the elimination of nodes and even whole subtrees, such that the enumeration of all possible nodes is avoided. The effort of solving problem P by a branch-and-bound method is determined by the percentage of eliminated nodes and the effort for solving the candidate subproblems.

In practice, lots of different criteria how to select the next candidate subproblem exist. Every criterion leads to a different enumeration of the search tree. The selection of an appropriate partition rule, e.g., the selection of a variable to branch on, heavily influences the computation time of a branch-and-bound method.

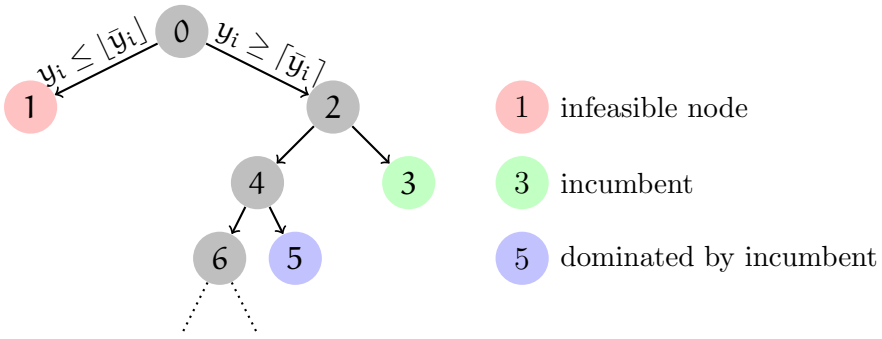

Fig. 2.3: Illustration of Branch-and-Bound Search Tree

After the general introduction of the branch-and-bound approach, we focus on NLP-based branch-and-bound methods for solving convex MINLP problems. Note, that these techniques can also be applied to solve strictly convex MIQP problems, see Section 2.11, since strictly convex MIQPs are a subclass of convex mixed-integer nonlinear optimization problems. NLP-based branch-and-bound methods rely on continuous nonlinear relaxations derived from the original mixed-integer nonlinear program by dropping the integrality condition on the integer variables, see problem (2.47).

The root of the search tree corresponds to the continuous relaxation of the original MINLP problem (1.6), i.e., NLP (2.47). Its solution is denoted by $(\bar{x}, \bar{y})$. If all integer variables possess integral values at $(\bar{x}, \bar{y})$ the whole process can be stopped, since the optimal solution for MINLP (1.6) is found. Otherwise, we perform a partition by branching on a fractional integer variable y_i with $\bar{y}_i \notin \mathbb{N}$ and splitting the relaxation into two disjoint problems of form (2.47). The upper bound on $y_i \in Y_{\mathbb{R}}$ is set to $\lfloor \bar{y}_i \rfloor$ for one problem, while the lower bound on $y_i \in Y_{\mathbb{R}}$ for the other problem is set to $\lceil \bar{y}_i \rceil$. Repeating this so-called branching process, yields a binary search tree where all nodes correspond to continuous nonlinear programs (2.47) possessing different box-constraints for integer variables defined by I_L^k and I_U^k, see (2.50) and (2.51). In each iteration k a convex continuous nonlinear optimization problem NLP_r^k (2.47) is solved yielding a solution $(\bar{x}^k, \bar{y}^k) \in X \times Y_{\mathbb{R}}^k$. The set $Y_{\mathbb{R}}^k$ is given by

$$Y_{\mathbb{R}}^k \;:=\; \{y \in Y_{\mathbb{R}} : y_i \geq (y_l^k)_i,\ i \in I_L^k \text{ and } y_i \leq (y_u^k)_i,\ i \in I_U^k\}. \tag{2.58}$$

A node of the branch-and-bound search tree can be fathomed according to Lemma 2.4 and Corollary 2.2.

An integer feasible optimal solution $(\bar{x}^k, \bar{y}^k) \in X \times Y$ of a subproblem, is an upper bound for the optimal objective function value of MINLP (1.6), see Criterion 3 in Corollary 2.2. The minimal objective function value over all nodes that are not explored, is a lower bound on the optimal objective function value of MINLP (1.6). Thus, whenever a feasible integer solution is found, an upper and a lower bound on the optimal solution of problem (1.6) is known.

The more often the fathoming rules stated in Corollary 2.2 can be applied, the smaller is the number of subproblems, that has to be enumerated. The enumeration process stops, if all nodes of the tree are fathomed. The optimal solution of MINLP (1.6) is the integer feasible solution with the lowest objective value. If no subproblem possesses an integer feasible solution, problem (1.6) is infeasible.

The general concept of branch-and-bound can only be applied, if valid lower bounds can be derived. Usually the relaxations are solved to global optimality and the global solution provides the corresponding lower bound. As a consequence, applying local continuous NLP solution techniques for solving non-convex relaxations of non-convex MINLP problems (1.1) can neither guarantee valid lower bounds nor global optimal solutions. In practice, local NLP techniques are often applied, since on the one hand the optimal solution of MINLP (1.1) is often found, see Chapter 6. On the other hand no efficient methods for solving non-convex NLP problems to global optimality or even for deriving lower bounds of sufficient quality are known up to now.

In Section 2.10 we review the theoretical concept of an algorithm developed by Exler and Schittkowski [45] for mixed-integer nonlinear optimization problems, whose implementation is called MISQP. As shown in Chapter 6, it is an efficient solution method for MINLP problems, i.e., it obtains feasible solutions with an objective value close to the global optimum within very few iteration, i.e., with very few function and gradient evaluations. To improve the robustness with respect to binary variables,

a special branching strategy was developed and implemented within a NLP-based branch-and-bound solver. Since binary variables are handled separately and the approach is designed to solve non-convex MINLP problems (1.1), we proceed from

$$\begin{aligned} & x \in X,\ y \in Y,\ b \in B: \\ & \quad \min \quad f(x,y,b) \\ & \quad \text{s.t.} \quad g_j(x,y,b) \ = \ 0, \quad j = 1,\dots,m_e, \\ & \qquad\quad g_j(x,y,b) \ \geq \ 0, \quad j = m_e+1,\dots,m. \end{aligned} \tag{2.59}$$

The number of binary variables $b \in B$ with

$$B \ := \ \mathbb{B}^{n_b} \ = \ \{0,1\}^{n_b}. \tag{2.60}$$

is denoted by n_b. The sets X and Y represent box constraints for the n_c continuous and n_i integer variables respectively, see (1.2).

The idea is to implement a NLP-based branch-and-bound method, where branching can be performed either subject to integer and binary variables or subject to the binary variables only. By relaxing binary variables we replace the set B by

$$B_{\mathbb{R}} \ := \ \{b \in \mathbb{R}^{n_b} : 0 \leq b_i \leq 1,\ i = 1,\dots,n_b\}. \tag{2.61}$$

In iteration k we can choose between solving continuous branch-and-bound subproblems derived from the continuous relaxation of MINLP (2.59), which are given by

$$\begin{aligned} & x \in X,\ y \in Y_{\mathbb{R}},\ b \in B_{\mathbb{R}}: \\ & \quad \min \quad f(x,y,b) \\ & \quad \text{s.t.} \quad g_j(x,y,b) \ = \ 0, \qquad j = 1,\dots,m_e, \\ & \qquad\quad g_j(x,y,b) \ \geq \ 0, \qquad j = m_e+1,\dots,m, \\ & \qquad\qquad y_i \ \geq \ (y_l^k)_i, \quad i \in I_L^k, \\ & \qquad\qquad y_i \ \leq \ (y_u^k)_i, \quad i \in I_U^k, \\ & \qquad\qquad b_i \ = \ 1, \qquad i \in I_L^k, \\ & \qquad\qquad b_i \ = \ 0, \qquad i \in I_U^k, \end{aligned} \tag{2.62}$$

where I_L^k and I_U^k are defined by (2.50) and (2.51). In this case the algorithm performs like a well-known NLP-based branch-and-bound method.

If branching is performed subject to the binary variables only, we solve in iteration k

a partly relaxed mixed-integer nonlinear program

$$
\begin{aligned}
x \in X,\ y \in Y,\ b \in B_{\mathbb{R}}: & \\
\min \quad & f(x,y,b) \\
\text{s.t.} \quad & g_j(x,y,b) = 0, \quad j = 1,\dots,m_e, \\
& g_j(x,y,b) \geq 0, \quad j = m_e+1,\dots,m, \\
& b_i = 1, \quad i \in I_L^k, \\
& b_i = 0, \quad i \in I_U^k,
\end{aligned}
\tag{2.63}
$$

with $I_L^k \cap B$ and $I_U^k \cap B$ defined by (2.50) and (2.51), instead. Therefore, we replace NLP subproblem (2.62) by MINLP subproblem (2.63). The application of a NLP-based branch-and-bound method requires, that the mixed-integer program (2.59) is relaxable subject to those variables that should be branch on. In our case, the problem has to be relaxable subject to either binary variables only or subject to all binary and integer variables. In both cases, the functions $f(x,y,b)$ and $g_j(x,y,b),\ j = 1,\dots,m$, must be continuously differentiable subject to the continuous and the relaxed variables in order to apply gradient-based solution methods.

Since NLP-based branch-and-bound methods require the solution of one continuous nonlinear program for each node of the branch-and-bound search tree, they can be inefficient in practice, see also Chapter 6. Furthermore, the solutions of a large number of nodes do not have a physical meaning, since the integer variables take fractional values, e.g., 2.5 machines are operated.

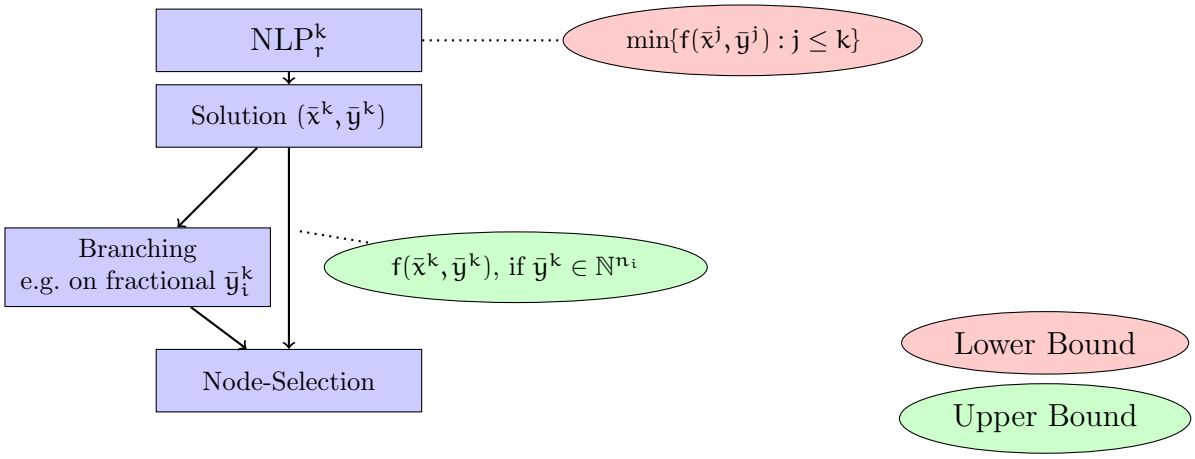

Fig. 2.4: NLP-based Branch-and-Bound

Figure 2.4 illustrates the algorithmic layout of a branch-and-bound iteration. The iterations starts by solving the relaxation RCS of the chosen candidate subproblem CS. If the corresponding solution is feasible for the original problem P an upper bound

is obtained. Otherwise we need to perform further partitioning, e.g., by branching on a fractional integer variable. Finally the next branch-and-bound node is selected, i.e., the next CS is chosen. The lower bound is given by the lowest objective value of all unexplored nodes.

2.5 Linear Outer Approximation

Linear outer approximation algorithms are often applied for solving the convex mixed-integer nonlinear program (1.6). We adopt the notation of the previous sections and assume that no linear equality constraints are contained in the convex MINLP (1.6), i.e., $m_e = 0$ and $\mathbb{J} = \mathbb{J}_>$.

Duran and Grossmann [43] as well as Fletcher and Leyffer [50] suggested linear outer approximation algorithms. The one proposed by Fletcher and Leyffer in [50] guarantees global optimality, if the following assumptions are satisfied.

Assumption 2.2. *1.* $f(x,y)$ *is convex and* $g_j(x,y),\ j = 1,\dots,m$ *are concave on* $X \times Y_{\mathbb{R}}$ *and the set* X *defined by* (1.2) *is nonempty and compact.*

2. $f(x,y)$ *and* $g_j(x,y),\ j = 1,\dots,m$ *are continuously differentiable on* $X \times Y_{\mathbb{R}}$.

3. The linear independent constraint qualification, stated in Definition 2.6, holds at each optimal solution of problem NLP(y*) and the corresponding nonlinear feasibility problem F(*y*) for all* $y \in Y$, *see* (2.52) *and* (2.54).

Note, that gradients need to be provided for all integer variables $y \in Y$, which might not be possible for some applications, e.g., if problem (1.6) is not relaxable and gradient information has to be approximated numerically at neighbored grid-points, see Exler, Lehmann and Schittkowski [47].

Part 1 and part 2 of Assumption 2.2 are also required for NLP-based branch-and-bound algorithms, see Section 2.4, to ensure that the subproblems can be solved efficiently to global optimality by a local NLP solver, e.g., the trust region method of Yuan, see Section 2.2. Part 3 of Assumption 2.2 might be violated in practice, but it is necessary to derive the algorithmic concept of linear outer approximation algorithms.

Linear outer approximation methods solve the convex MINLP problem (1.6) iteratively. In each iteration k a continuous NLP subproblem is formulated by fixing the integer variables $y \equiv y^k$ in MINLP (1.6). Based on the corresponding solution $\bar{x}_{y^k}$ a mixed-integer linear master program similar to $MILP_r^k$ (2.55) is set up by linearizing the constraints and the objective function at $(\bar{x}_{y^k}, y^k)$. Its solution provides a lower bound on the optimal objective value of MINLP (1.6) and the new integer value y^{k+1}. Therefore, each iteration k corresponds to exactly one integer value $y^k \in Y$.

Linear outer approximation algorithms possess two essential properties. On the one hand, they terminate finitely, as long as Y is of finite dimension, since previously

considered integer values are not explored twice and their number is finite. On the other hand, global optimality is ensured, as the master problem is a continuously improving linear relaxation of the original convex MINLP (1.6), see Fletcher and Leyffer [50].

Both properties are guaranteed by the integration of additional linearizations of the original constraints and possibly of the objective function in the master problem. In iteration k these linearizations are constructed, such that they are violated by y^k for any value of $x \in X$. The same holds for any previous integer iterate y^i with $i \leq k$, where y^i denotes the integer value in iteration i. To derive these linearizations, we distinguish between feasible and infeasible integer values $y^k \in Y$. If y^k is a feasible integer value, i.e.,

$$\{x \in X: \quad g_j(x, y^k) \ \geq \ 0, \ \ \forall\, j \in \mathbb{J}\,\} \ \neq \ \emptyset, \tag{2.64}$$

it is part of the set V defined by

$$V := \{y \in Y : \exists\, x \in X \text{ with } g_j(x, y) \geq 0, \ \forall\, j \in \mathbb{J}\}. \tag{2.65}$$

The set V contains all feasible integer values, i.e., for each $y \in V$ there exists at least one $x \in X$ that satisfies the constraints of (1.6).

If y^k is an infeasible integer value, the constraints of MINLP (1.6) are violated independent of the value of the continuous variables $x \in X$, i.e.,

$$\left\{x \in X : g_j(x, y^k) \geq 0, \ \forall\, j \in \mathbb{J}\right\} = \emptyset. \tag{2.66}$$

The linearizations, that should be included in the master problem, are derived from the solution of the continuous nonlinear program, which is solved in each iteration. For some integer iterate $y^k \in Y$ we proceed with the continuous nonlinear program NLP(y^k), given by

$$\begin{aligned} x \in \mathbb{R}^{n_c}: & \\ \min \quad & f(x, y^k) \\ \text{s.t.} \quad & g_j(x, y^k) \ \geq \ 0, \ \ j \in \mathbb{J}, \end{aligned} \tag{2.67}$$

in the corresponding iteration k. This problem is equivalent to NLP (2.52), introduced in Section 2.3. If $y^k \in V$ holds, then the existence of the optimal solution of NLP(y^k) denoted by $\bar{x}_{y^k}$ is guaranteed by Assumption 2.2, see Fletcher and Leyffer [50].

If NLP(y^k) is infeasible, i.e., $y^k \in Y \backslash V$, we search for a value of $x \in X$ minimizing the constraint violation by solving a feasibility problem. The corresponding continuous NLP is denoted by F(y^k) and we consider the subsequent formulation. See Fletcher and Leyffer [50] for a more general formulation.

$$\begin{aligned} & x \in \mathbb{R}^{n_c}, \ \eta \in \mathbb{R}_+: \\ & \min \quad \eta \\ & \text{s.t.} \quad g_j(x, y^k) + \eta \ \geq \ 0, \ \ \forall j \in \mathbb{J}. \end{aligned} \tag{2.68}$$

We denote the solution of $F(y^k)$ by $(\bar{x}_{y^k}, \bar{\eta})$.

We define the set T containing all feasible integer values $y \in Y$ and the corresponding solution $\bar{x}_y$ of the nonlinear subproblem NLP(y), i.e.,

$$T := \{(\bar{x}_y, y) \in X \times Y : \bar{x}_y \text{ is an optimal solution of NLP}(y)\}. \tag{2.69}$$

Analogue to T, the set S associated with infeasible integer values y and the corresponding optimal value $\bar{x}_y$ of $F(y)$ is introduced, i.e.,

$$S := \{(\bar{x}_y, y) \in X \times Y \backslash V : \bar{x}_y \text{ is part of the optimal solution of } F(y)\}. \tag{2.70}$$

Every integer value $y^k \in Y$ is either contained in T or in S.

Since the sets T and S are not known a priori, they are approximated dynamically. In iteration k the sets T and S are replaced by

$$T^k := \left\{(\bar{x}_{y^i}, y^i) \in X \times \{y^1, \ldots, y^k\} \cap V : \bar{x}_{y^i} \text{ solves NLP}(y^i)\right\} \tag{2.71}$$

$$S^k := \left\{(\bar{x}_{y^j}, y^j) \in X \times \{y^1, \ldots, y^k\} \cap Y \backslash V : \bar{x}_{y^j} \text{ solves } F(y^j)\right\}, \tag{2.72}$$

containing the solutions of NLP(y^i) with $i \leq k$ and $F(y^j)$ with $j \leq k$ of all previous iterations. The sets T^k and S^k are subsets of the sets T and S.

Now we derive the mixed-integer linear master problem, which contains linearizations of the objective function at the solutions $(\bar{x}_{y^i}, y^i)$ of NLP(y^i), i.e., $y^i \in V$, of all previous iterates $i \leq k$. In iteration k the linearization at $(\bar{x}_{y^k}, y^k)$ with $y^k \in V$ is given by

$$f(\bar{x}_{y^k}, y^k) + \nabla_{x,y} f(\bar{x}_{y^k}, y^k)^T \begin{pmatrix} x - \bar{x}_{y^k} \\ y - y^k \end{pmatrix}, \tag{2.73}$$

for some $(x, y) \in X \times Y$. $\nabla_{x,y} f(x, y)$ denotes the gradient of the objective function $f(x, y)$.

In addition to (2.73) the master problem also contains linearizations of those constraints, that are strongly active, see Definition 2.5, at the solution $(\bar{x}_{y^k}, y^k)$ of NLP(y^k) or $F(y^k)$. They are given by

$$g(\bar{x}_{y^k}, y^k) + [\nabla_{x,y} g(\bar{x}_{y^k}, y^k)]^T \begin{pmatrix} x - \bar{x}_{y^k} \\ y - y^k \end{pmatrix} \geq 0, \tag{2.74}$$

for some $(x, y) \in X \times Y$. $[\nabla_{x,y} g(x, y)]$ denotes the Jacobian matrix of the constraints $g_j(x, y), \quad j \in \mathbb{J}$. The mixed-integer linear master problems are denoted by $MILP_r^k$, since they provide a linear relaxation of problem (1.6). The quality of the relaxation is improved in each iteration by an increasing number of constraints obtained from (2.73) and (2.74). The aim of the master problem is to find a promising unexplored integer value or to detect that there is none. If the master problem is infeasible, either the global optimal solution of MINLP (1.6) is obtained subject to a certain termination

accuracy or it is detected that the feasible region is empty. According to Fletcher and Leyffer [50] the master problem in iteration k defined by

$$
\begin{aligned}
& x \in \mathbb{R}^{n_c},\ y \in \mathbb{N}^{n_i},\ \eta \in \mathbb{R}: \qquad (2.75)\\
& \min \quad \eta \\
& \text{s.t.} \quad f(\bar{x}_{y^i}, y^i) + \nabla_{x,y} f(\bar{x}_{y^i}, y^i)^T \begin{pmatrix} x - \bar{x}_{y^i} \\ y - y^i \end{pmatrix} \leq \eta, \quad \forall (\bar{x}_{y^i}, y^i) \in T^k, \\
& \qquad g(\bar{x}_{y^i}, y^i) + [\nabla_{x,y} g(\bar{x}_{y^i}, y^i)]^T \begin{pmatrix} x - \bar{x}_{y^i} \\ y - y^i \end{pmatrix} \geq 0, \quad \forall (\bar{x}_{y^i}, y^i) \in T^k, \\
& \qquad g(\bar{x}_{y^j}, y^j) + [\nabla_{x,y} g(\bar{x}_{y^j}, y^j)]^T \begin{pmatrix} x - \bar{x}_{y^j} \\ y - y^j \end{pmatrix} \geq 0, \quad \forall (\bar{x}_{y^j}, y^j) \in S^k, \\
& \qquad \eta \leq \hat{\eta}^k - \varepsilon.
\end{aligned}
$$

We denote it MILP($T^k, S^k, \hat{\eta}^k, \varepsilon$), since it depends on the sets T^k and S^k, an appropriate upper bound $\hat{\eta}^k$ and a termination accuracy ε. $(\bar{x}_{y^i}, y^i) \in T^k$ with $i \leq k$ and $y^i \in V$ denote the solution of NLP(y^i) in the previous iterations, while $(\bar{x}_{y^j}, y^j) \in S^k$ with $j \leq k$ and $y^j \in Y \backslash V$ denote the solutions of F(y^j) in the previous iterations. $\hat{\eta}^k$ defines an appropriate upper bound on η and the condition

$$
\begin{aligned}
\eta &\leq \hat{\eta}^k - \varepsilon, \text{ with} \\
\hat{\eta}^k &:= \min\{f(\bar{x}_{y^i}, y^i) : (\bar{x}_{y^i}, y^i) \in T^k\},
\end{aligned}
\qquad (2.76)
$$

ensures that feasible solutions $(\bar{x}_{y^i}, y^i) \in T^k$ of previous iterations are infeasible in (2.75).

Now we review the theoretical background of the linear outer approximation algorithm according to Fletcher and Leyffer [50]. The optimal solution $(x^*, y^*) \in X \times Y$ of the convex MINLP (1.6) is equivalent to the solution of NLP(y^*) with $y^* \in V$ given by (2.65). As a consequence, MINLP (1.6) is equivalent to problem

$$
\min_{y \in V} \{ \text{NLP}(y) \}. \qquad (2.77)
$$

According to Fletcher and Leyffer [50] Assumption 2.2 ensures, that problem (2.77) possesses the same objective value as

$$
\begin{aligned}
\min_{y \in V} \quad & \min_{x \in X} \quad f(\bar{x}_y, y) + \nabla_{x,y} f(\bar{x}_y, y)^T \begin{pmatrix} x - \bar{x}_y \\ 0 \end{pmatrix} \\
& \text{s.t.} \quad g(\bar{x}_y, y) + [\nabla_{x,y} g(\bar{x}_y, y)]^T \begin{pmatrix} x - \bar{x}_y \\ 0 \end{pmatrix} \geq 0.
\end{aligned}
\qquad (2.78)
$$

Note, that the inner optimization problem of (2.78) is always bounded, due to the compactness of the set X and the equivalence to problem (2.77), see Fletcher and Leyffer [50]. The number of constraints of problem (2.78) can be reduced, such that

only strongly active constraints at the optimal solution $(\bar{x}_y, y)$ of NLP(y), see Definition 2.5, are included, see Fletcher and Leyffer [50]. After introducing an artificial variable $\eta \in \mathbb{R}$, that minimizes the linearizations of the objective function, problem (2.78) becomes

$$
\begin{aligned}
& x \in \mathbb{R}^{n_c},\ y \in \mathbb{N}^{n_i},\ \eta \in \mathbb{R}: \qquad (2.79)\\
& \min \quad \eta \\
& \text{s.t.} \quad f(\bar{x}_{y^i}, y^i) + \nabla_{x,y}\, f(\bar{x}_{y^i}, y^i)^T \begin{pmatrix} x - \bar{x}_{y^i} \\ y - y^i \end{pmatrix} \leq \eta, \quad \forall (\bar{x}_{y^i}, y^i) \in T, \\
& \qquad g(\bar{x}_{y^i}, y^i) + [\nabla_{x,y}\, g(\bar{x}_{y^i}, y^i)]^T \begin{pmatrix} x - \bar{x}_{y^i} \\ y - y^i \end{pmatrix} \geq 0, \quad \forall (\bar{x}_{y^i}, y^i) \in T, \\
& \qquad g(\bar{x}_{y^j}, y^j) + [\nabla_{x,y}\, g(\bar{x}_{y^j}, y^j)]^T \begin{pmatrix} x - \bar{x}_{y^j} \\ y - y^j \end{pmatrix} \geq 0, \quad \forall (\bar{x}_{y^j}, y^j) \in S.
\end{aligned}
$$

Due to previous considerations MILP (2.79) has the same objective value as the convex MINLP (1.6), see Bonami et. al. [27]. Due to the definition of the sets T and S in (2.69) and (2.70), one has to solve problem NLP(y), or F(y) respectively, for all $y \in Y$ to be able to set up problem (2.79). Note, that both sets T and S are finite due to the finiteness of Y, see (1.2).

The subsequent algorithm reviews the linear outer approximation algorithm according to Fletcher and Leyffer [50].

Algorithm 2.2. *1. Let $x^0 \in X$ and $y^0 \in Y$ be starting values. Define the sets $T^{-1} = S^{-1} = \emptyset$. Define the optimality tolerance $\varepsilon_{OA} > 0$ and set $k := 0$.*

Initialize best known solution by $(x^*, y^*) := (x^0, y^0)$ and define $f^* := \infty$.

Evaluate the functions $f(x^0, y^0)$ and $g(x^0, y^0)$ and determine gradients $\nabla_{x,y} f(x^0, y^0)$ and $\nabla_{x,y} g(x^0, y^0)$.

2. **If** $y^k \in Y$ *given by* (2.65),
then *solve NLP(y^k) defined by* (2.67). *Denote the solution by* $(\bar{x}_{y^k}, y^k)$.
If $f(\bar{x}_{y^k}, y^k) < f^*$,
then *update* $f^* := f(\bar{x}_{y^k}, y^k)$ *and* $(x^*, y^*) := (\bar{x}_{y^k}, y^k)$.

Else *solve F(y^k) defined by* (2.68) *and denote the solution by* $(\bar{x}_{y^k}, y^k)$.

3. **If** $y^k \in V$,
then *update set* T^{k-1} *defined by* (2.71)*:*

$$T^k := T^{k-1} \cup \{(\bar{x}_{y^k}, y^k)\}, \qquad (2.80)$$

$$S^k := S^{k-1}. \qquad (2.81)$$

Else *update set* S^{k-1} *defined by* (2.72)*:*

$$T^k := T^{k-1}, \tag{2.82}$$

$$S^k := S^{k-1} \cup \{(\bar{x}_{y^k}, y^k)\}. \tag{2.83}$$

Solve linear outer approximation master problem MILP($T^k, S^k, f^*, \varepsilon_{OA}$*) given by* (2.75)*.*

If *the master problem* (2.75) *is feasible,*
then *denote the solution by* (x^{k+1}, y^{k+1})*.*
Set $k := k + 1$ *and* **GOTO** *Step 2.*
Else **Stop** .

The work of Fletcher and Leyffer [50] yields the following theorem.

Theorem 2.4. *If Assumption 2.2 holds, then Algorithm 2.2 terminates after a finite number of iterations yielding an optimal solution of the convex MINLP* (1.6) *or it detects, that MINLP* (1.6) *is infeasible.*

Proof. See Fletcher and Leyffer [50] and note, that the set Y given by (1.2) is finite by definition. □

Theorem 2.4 is based on the observation that no integer value is generated twice by Algorithm 2.2. This property is established by the subsequent lemma for infeasible integer values $y \notin V$, where V is defined by (2.65).

Lemma 2.5. *Consider* $y^k \in Y \backslash V$ *with* V *given by* (2.65) *and let* $(\bar{x}_{y^k}, \bar{\eta})$ *be the optimal solution of F(*y^k*) given by* (2.68) *with* $\bar{\eta} > 0$*.*

Then y^k *does not satisfy the subsequent constraints for any value* $x \in X$*.*

$$g(\bar{x}_{y^k}, y^k) + [\nabla_{x,y} g(\bar{x}_{y^k}, y^k)]^T \begin{pmatrix} x - \bar{x}_{y^k} \\ y - y^k \end{pmatrix} \geq 0, \quad \forall j \in \mathbb{J}. \tag{2.84}$$

Proof. See Fletcher and Leyffer [50]. □

The same holds for feasible integer values $y \in V$.

Lemma 2.6. *Let Assumptions 2.2 hold and let NLP(*y^k*) be feasible, where* $\bar{x}_{y^k}$ *is its optimal solution. Then* y^k *does not satisfy the subsequent constraints for any value of* $x \in X$*.*

$$\eta < f(\bar{x}_{y^k}, y^k), \tag{2.85}$$

$$f(\bar{x}_{y^k}, y^k) + \nabla_{x,y} f(\bar{x}_{y^k}, y^k)^T \begin{pmatrix} x - \bar{x}_{y^k} \\ y - y^k \end{pmatrix} \leq \eta, \tag{2.86}$$

$$g_j(\bar{x}_{y^k}, y^k) + \nabla_{x,y} g_j(\bar{x}_{y^k}, y^k)^T \begin{pmatrix} x - \bar{x}_{y^k} \\ y - y^k \end{pmatrix} \geq 0, \quad \forall j \in \mathbb{J}. \tag{2.87}$$

Proof. Note, that the subsequent proof is extracted from Fletcher and Leyffer [50].

Assume, that there exists a point $(\hat{\eta}, \hat{x}, y^k)$ satisfying (2.85), (2.86) and (2.87). Since $\bar{x}_{y^k}$ is the optimal solution of NLP(y^k) and a constraint qualification holds due to Assumptions 2.2, no feasible descent direction $\hat{x} - \bar{x}_{y^k}$ exits, i.e.,

$$\begin{aligned} 0 &\leq g(\bar{x}_{y^k}, y^k) + [\nabla_{x,y} g(\bar{x}_{y^k}, y^k)]^T \begin{pmatrix} \hat{x} - \bar{x}_{y^k} \\ 0 \end{pmatrix} \\ \Rightarrow \quad 0 &\leq \nabla_{x,y} f(\bar{x}_{y^k}, y^k)^T \begin{pmatrix} \hat{x} - \bar{x}_{y^k} \\ 0 \end{pmatrix}, \end{aligned} \tag{2.88}$$

holds for all $\hat{x} \in X$. Substitution of (2.88) into (2.86) yields

$$\hat{\eta} \geq f(\bar{x}_{y^k}, y^k), \tag{2.89}$$

which contradicts (2.85). □

Now, we will motivate the use of the trust region method of Yuan, i.e., Algorithm 2.1, within a linear outer approximation method such as Algorithm 2.2. As a consequence we introduce an additional iteration counter l associated with the trust region method of Yuan, while the iteration index k corresponds to the linear outer approximation method. Therefore, we propose to apply Algorithm 2.1 for solving NLP(y^k) for some fixed $y^k \in Y$.

As stated in Lemma 2.2, the trust region Algorithm 2.1 might converge towards an infeasible stationary point of NLP(y^k) with $y^k \in Y$, see Definition 2.10. The subsequent corollaries show, that this property can be exploited within a linear outer approximation method as we need not distinguish between solving NLP(y^k) given by (2.67) and the feasibility problem F(y^k) for some fixed $y^k \in Y$. First, we review the relation of an infeasible stationary point specified in Definition 2.10 and the solution of feasibility problem (2.68).

Corollary 2.3. *Let $\bar{x}_{y^k}$ be an infeasible stationary point specified in Definition 2.10. Then $\bar{x}_{y^k}$ is the optimal solution of feasibility problem F(y^k) given by* (2.68) *for any fixed $y^k \in Y \backslash V$.*

Proof. See Yuan [112] or Jarre and Stoer [66]. □

Now we are going to show, that the trust region Algorithm 2.1 converges towards the solution of NLP(y^k), if $y^k \in V$. Otherwise, i.e., $y^k \in Y \backslash V$, Algorithm 2.1 converges towards the optimal solution of feasibility problem F(y^k) (2.68). This implies, that we need not distinguish between solving problem NLP(y^k) given by (2.67) and F(y^k) given by (2.68) for fixed $y^k \in Y$. As we establish in the subsequent corollary, the infeasibility of NLP(y^k) with $y^k \in Y \backslash V$ is recognized by obtaining the optimal solution of (2.68), where V is defined by (2.65).

For this task we introduce the subsequent linear program, which is set up at some fixed $(x^l, y^k) \in X \times Y$ and therefore denoted by LPF(x^l, y^k)

$$\begin{aligned} & (d_F)_x \in \mathbb{R}^{n_c},\ \eta \in \mathbb{R}_+ : \\ & \min \quad \eta \\ & \text{s.t.} \quad g_j(x^l, y^k) + \nabla_x g_j(x^l, y^k)^T (d_F)_x + \eta \ \geq \ 0, \quad \forall j \in \mathbb{J}. \end{aligned} \tag{2.90}$$

The solution of LPF(x^l, y^k) given by (2.90) is denoted by $((d_F^l)_x, \eta^l)$.

Corollary 2.4. *Let $((d_F^l)_x, \eta^l)$ be the optimal solution of LPF(x^l, y^k) given by* (2.90)*. Then the KKT-conditions of LPF(x^l, y^k) given by* (2.90) *coincide with those of feasibility problem F(y^k) given by* (2.68) *for any fixed $y^k \in Y$, if $(d_F^l)_x = 0$ holds.*

Proof. A KKT point $(\bar{\eta}, \bar{x}_y, (\bar{\lambda}_F, \bar{\lambda}_\eta))$ of problem (2.68), with

$$\bar{\lambda}_F \ := \ \begin{pmatrix} \bar{\lambda}_1 \\ \vdots \\ \bar{\lambda}_m \end{pmatrix},$$

where $\bar{\lambda}_F \in \mathbb{R}^m$ denotes the Lagrangian multiplier associated with the constraints $g_j,\ j \in \mathbb{J}$, and $\bar{\lambda}_\eta$ denotes the Lagrangian multiplier of the non-negativity condition on η, satisfies the subsequent KKT-conditions:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} - \sum_{j \in \mathbb{J}} (\bar{\lambda}_F)_j \begin{pmatrix} \nabla_x g_j(\bar{x}_y, y) \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ \bar{\lambda}_\eta \end{pmatrix} \ = \ 0, \tag{2.91}$$

$$(g_j(\bar{x}_y, y) + \bar{\eta})\,(\bar{\lambda}_F)_j \ = \ 0,\ \forall j \in \mathbb{J}, \tag{2.92}$$

$$\bar{\eta}\bar{\lambda}_\eta \ = \ 0, \tag{2.93}$$

$$g_j(\bar{x}_y, y) + \bar{\eta} \ \geq \ 0,\ \forall j \in \mathbb{J}, \tag{2.94}$$

$$\bar{\eta} \ \geq \ 0, \tag{2.95}$$

$$(\bar{\lambda}_F)_j \ \geq \ 0,\ \forall j \in \mathbb{J}, \tag{2.96}$$

$$\bar{\lambda}_\eta \ \geq \ 0. \tag{2.97}$$

For LPF(x^l, y^k) given by (2.90) the KKT-conditions for a KKT-point $(\eta^l, (d_F^l)_x, (\lambda_F^l, \lambda_\eta^l))$

yield

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} - \sum_{j \in \mathbb{J}} (\lambda_F^l)_j \begin{pmatrix} \nabla_x g_j(x^l, y^k) \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ \lambda_\eta^l \end{pmatrix} = 0, \tag{2.98}$$

$$\left(g_j(x^l, y^k) + \nabla_x g_j(x^l, y^k)^T (d_F^l)_x + \eta^l\right) (\lambda_F^l)_j = 0, \ \forall j \in \mathbb{J}, \tag{2.99}$$

$$\eta^l \lambda_\eta^l = 0, \tag{2.100}$$

$$g_j(x^l, y^k) + \nabla_x g_j(x^l, y^k)^T (d_F^l)_x + \eta^l \geq 0, \ \forall j \in \mathbb{J}, \tag{2.101}$$

$$\eta^l \geq 0, \tag{2.102}$$

$$(\lambda_F^l)_j \geq 0, \ \forall j \in \mathbb{J}, \tag{2.103}$$

$$\lambda_\eta^l \geq 0, \tag{2.104}$$

where λ_η^l denotes the Lagrangian multiplier associated with the non-negativity condition on η, while $\lambda_F^l \in \mathbb{R}^m$ denotes the Lagrangian multipliers of the linearizations of the constraints g_j, $j \in \mathbb{J}$. For $(d_F^l)_x = 0$ the KKT-conditions of $F(y^k)$ (2.91) - (2.97) and $LPF(x^l, y^k)$ (2.98) - (2.104) are equivalent with $\bar{x}_y := x^l$, $\bar{\lambda}_F := \lambda_F^l$, $\bar{\lambda}_\eta := \lambda_\eta^l$ and $\bar{\eta} := \eta^l$. □

As a consequence, solving $LPF(x^l, y^k)$ given by (2.90) allows to identify infeasible stationary points, specified in Definition 2.10. Specifically, $\bar{x}_y$ is an infeasible stationary point of $NLP(y^k)$, which is the solution of problem $F(y^k)$ (2.68), if the solution $((d_F^l)_x, \eta^l)$ of $LPF(x^l, y^k)$ with $x^l = \bar{x}_y$ satisfies $(d_F^l)_x = 0$ and $\eta^l > 0$.

Corollary 2.5. *Let Assumptions 2.1 and Assumptions 2.2 hold. Furthermore, apply Yuan's trust region Algorithm 2.1 for solving NLP(y^k) given by* (2.67) *with* $y^k \in Y$*, e.g., arising in Step 2 of the linear outer approximation Algorithm 2.2.*

If $y^k \in V$*, where* V *is given by* (2.65)*, then the iteration sequence created by Algorithm 2.1 converges towards a stationary point of NLP(y^k).*

If $y^k \in Y \backslash V$*, then the iteration sequence converges towards an infeasible stationary point of NLP(y^k) introduced in Definition 2.10, which is the optimal solution of feasibility problem F(y^k)* (2.68) *due to Corollary 2.3.*

Proof. Due to Assumptions 2.2, there exists no singular stationary point introduced in Definition 2.11 for any continuous nonlinear program derived from MINLP (1.6) by fixing the integer variables $y^k \in Y$, see Yuan [112] or Jarre and Stoer [66]. If $y^k \in Y \backslash V$, then no stationary point according to Definition 2.12 of $NLP(y^k)$ exists. Since Assumption 2.1 holds, Algorithm 2.1 converges towards an infeasible stationary point introduced in Definition 2.10 due to Lemma 2.2.

If $y^k \in V$, then no infeasible stationary point exists due to the subsequent computations: Since MINLP (1.6) is convex, the constraints $g_j,\ j \in \mathbb{J}$ are concave, i.e.,

$$g_j((x^l, y^k) + d_x) \ \leq \ g_j(x^l, y^k) + \nabla_x g_j(x^l, y^k)^T d_x, \quad \forall j \in \mathbb{J} \tag{2.105}$$

holds for each $d_x \in \mathbb{R}^{n_c}$ and $(x^l, y^k) \in X \times V$. Since $y^k \in V$, there exists $\tilde{x} \in X$ with

$$g_j(\tilde{x}, y^k) \ \geq \ 0, \quad \forall j \in \mathbb{J}. \tag{2.106}$$

We prove by contradiction, that no infeasible stationary point exists. Assume therefore, that $(\hat{x}_y, y^k)$ is an infeasible stationary point of NLP(y^k) specified in Definition 2.10 and therefore an optimal solution of problem (2.68). As a consequence, the optimal solution $((d_F^l)_x, \eta^l)$ of LPF(x^k, y^k) with $x^l = \hat{x}_y$ given by (2.90) satisfies $(d_F^l)_x = 0$ and $\eta^l > 0$ according to Corollary 2.4 and Definition 2.10. Consider now the search direction $(\bar{d}_F)_x \in \mathbb{R}^{n_c}$ given by

$$(\bar{d}_F)_x \ := \ \tilde{x} - \hat{x}_y \ \neq \ 0. \tag{2.107}$$

The constraints of LPF(x^l, y^k) with $x^l = \hat{x}_y$ given by (2.90) are satisfied for $(d_F)_x := (\bar{d}_F)_x$ and $\bar{\eta} \in \mathbb{R}_+$ with $\bar{\eta} := 0$, since

$$\begin{aligned} g_j(\hat{x}_y, y^k) + \nabla_x g_j(\hat{x}_y, y^k)^T (\bar{d}_F)_x + \bar{\eta} \ &\geq \ g_j((\hat{x}_y + (\bar{d}_F)_x), y^k) + \bar{\eta}, \\ &= \ g_j(\tilde{x}, y^k) + \bar{\eta}, \\ &\geq \ 0, \end{aligned} \tag{2.108}$$

holds $\forall j \in \mathbb{J}$, due to conditions (2.105) and (2.106). This contradicts the assumption, since $((\bar{d}_F)_x, \bar{\eta})$ with $\bar{\eta} = 0$ improves the optimal solution $(0, \eta^l)$ with $\eta^l = \|g(\hat{x}_y, y^k)^-\|_\infty > 0$ of LPF(x^l, y^k) with $x^l = \hat{x}_y$ given by (2.90).

As no infeasible stationary point exists for $y^k \in V$ and Assumption 2.1 holds, Algorithm 2.1 has to generate a bounded sequence of penalty parameters and therefore it converges towards a stationary point of NLP(y^k), since Theorem 2.3 holds. This proves the Corollary. □

The linear outer approximation approach is a well-known solution method for convex MINLP problems. The mixed-integer linear master problem (2.75) is a linear relaxation of the original convex MINLP (1.6). Since all linear relaxations remain valid during the whole solution process, they provide a global approximation. As a consequence, the performance for non-convex problems is rather poor. The reason is that parts of the feasible region, that often contain the global solution of a non-convex MINLP (1.1), are cut off. As shown by Fletcher and Leyffer [50], the efficiency of linear outer approximation algorithms in terms of the number of function evaluations is rather low since second order information is not included for integer variables.

Applying the linear outer approximation algorithm for a non-convex MINLP (1.1) causes some difficulties. On the one hand, it is very likely, that the optimal solution

of the non-convex MINLP is cut off by invalid linearizations of some non-convex functions. On the other hand, the continuous nonlinear programs NLP(y) and F(y) can not be solved to global optimality by efficient local NLP solvers.

For solving non-convex MINLP problems one can apply global optimization techniques, e.g., based on convex envelopes, see McCormick [80]. These techniques rely on knowledge of the problem structure, i.e., it is required that the problem functions are explicitly given. Furthermore, the computational effort is usually very high. If we nevertheless apply linear outer approximation instead of a global solver, e.g., BARON developed by Sahinidis [92], there is no guarantee to find the optimal solution, i.e., linear outer approximation becomes a heuristic approach.

For non-convex problems (1.1) Viswanathan and Grossmann [104] introduce a modified master problem. The original master problem (2.75) is extended by slack variables relaxing the linearizations of the constraints and the objective function. These additional variables are penalized in the objective function, in order to obtain a solution, where the slack values are as small as possible. Moreover, the nonlinear equality constraints are relaxed and linearized as inequalities. The corresponding orientation is determined by the sign of the Lagrangian multiplier at the solution of either NLP(y^k) or F(y^k) in iteration k. Therefore, the modified master problem is given by

$$
\begin{aligned}
&x \in X,\ y \in Y,\ \eta \in \mathbb{R},\\
&(s_l^i)_> \in \mathbb{R}_+,\ (s_l^i)_= \in \mathbb{R}_+,\ \forall i \in \hat{T}^k,\ \forall l \in \mathbb{J},\\
&(s_l^j)_> \in \mathbb{R}_+,\ (s_l^j)_= \in \mathbb{R}_+,\ \forall j \in \hat{S}^k,\ \forall l \in \mathbb{J}: \qquad (2.109)
\end{aligned}
$$

$$
\begin{aligned}
\min\ & \eta + \sum_{i \in \hat{T}^k,\ l \in \mathbb{J}_>} w_l^i (s_l^i)_> + \sum_{i \in \hat{T}^k,\ l \in \mathbb{J}_=} w_l^i (s_l^i)_= \\
& + \sum_{j \in \hat{S}^k,\ l \in \mathbb{J}_>} w_l^j (s_l^j)_> + \sum_{j \hat{S}^k,\ l \in \mathbb{J}_=} w_l^j (s_l^j)_=
\end{aligned}
$$

$$
\begin{aligned}
\text{s.t.}\quad & f(\bar{x}_{y^i}, y^i) + \nabla_{x,y}\, f(\bar{x}_{y^i}, y^i)^T \begin{pmatrix} x - \bar{x}_{y^i} \\ y - y^i \end{pmatrix} \le \eta, && \forall (\bar{x}_{y^i}, y^i) \in T^k,\\
& g_l(\bar{x}_{y^i}, y^i) + \nabla_{x,y}\, g_l(\bar{x}_{y^i}, y^i)^T \begin{pmatrix} x - \bar{x}_{y^i} \\ y - y^i \end{pmatrix} \ge -(s_l^i)_>, && \forall (\bar{x}_{y^i}, y^i) \in T^k,\\
& && \forall i \in \hat{T}^k,\ \forall l \in \mathbb{J}_>,\\
& \mathbb{T}_l^i \nabla_{x,y}\, g_l(\bar{x}_{y^i}, y^i)^T \begin{pmatrix} x - \bar{x}_{y^i} \\ y - y^i \end{pmatrix} \ge -(s_l^i)_=, && \forall (\bar{x}_{y^i}, y^i) \in T^k,\\
& && \forall i \in \hat{T}^k,\ \forall l \in \mathbb{J}_=,\\
& g_l(\bar{x}_{y^j}, y^j) + \nabla_{x,y}\, g_l(\bar{x}_{y^j}, y^j)^T \begin{pmatrix} x - \bar{x}_{y^j} \\ y - y^j \end{pmatrix} \ge -(s_l^j)_>, && \forall (\bar{x}_{y^j}, y^j) \in S^k,\\
& && \forall j \in \hat{S}^k,\ \forall l \in \mathbb{J}_>,\\
& \mathbb{T}_l^j \nabla_{x,y}\, g_l(\bar{x}_{y^j}, y^j)^T \begin{pmatrix} x - \bar{x}_{y^j} \\ y - y^j \end{pmatrix} \ge -(s_l^j)_=, && \forall (\bar{x}_{y^j}, y^j) \in S^k,\\
& && \forall j \in \hat{S}^k,\ \forall l \in \mathbb{J}_=.
\end{aligned}
$$

The sets $\hat{T}^k$ and $\hat{S}^k$ contain the iteration indices up to the current iteration k, where the integer value $y^j,\ j \le k$ was either feasible or infeasible. $w_l^i,\ w_l^j$ with $i \in \hat{S}^k,\ j \in \hat{T}^k,\ l \in$

$\mathbb{J}$ are positive weights, chosen to be a sufficiently large multiple of the value of the Lagrangian multipliers to penalize the slack variables. $\mathbb{T}_l^i$, $\mathbb{T}_l^j$ with $l \in \mathbb{J}_=$, $i \in \hat{T}^k$, $j \in \hat{S}^k$ are determined by the value of the Lagrangian multiplier of the equality constraints $g_l(\bar{x}^i, y^i)$ with $(\bar{x}_{y^i}, y^i) \in T^k$, $l \in \mathbb{J}_=$ and $g_l(\bar{x}^j, y^j)$ with $(\bar{x}_{y^j}, y^j) \in S^k$, $l \in \mathbb{J}_=$:

$$\mathbb{T}_l^j = \begin{cases} 1, & \text{if } \bar{\lambda}_l^j < 0 \\ -1, & \text{if } \bar{\lambda}_l^j > 0 \\ 0, & \text{if } \bar{\lambda}_l^j = 0 \end{cases} \quad \forall j \in \hat{S}^k \text{ and } l \in \mathbb{J}_= \tag{2.110}$$

$$\mathbb{T}_l^i = \begin{cases} 1, & \text{if } \bar{\lambda}_l^i < 0 \\ -1, & \text{if } \bar{\lambda}_l^i > 0 \\ 0, & \text{if } \bar{\lambda}_l^i = 0 \end{cases} \quad \forall i \in \hat{T}^k \text{ and } l \in \mathbb{J}_= \tag{2.111}$$

$\bar{\lambda}_l^i$ denotes the Lagrangian multiplier associated with constraint $l \in \mathbb{J}$ and the solution $(\bar{x}_{y^i}, y^i)$ see Viswanathan and Grossmann [104] for further details. If problem (2.109) is solved instead of master problem (2.75), no termination criterion is available. Floudas [52] suggests to stop, if in iteration k the optimal solution of NLP(y^k) is worse than the best previously found solution. This stopping criterion may lead to a premature termination of the algorithm.

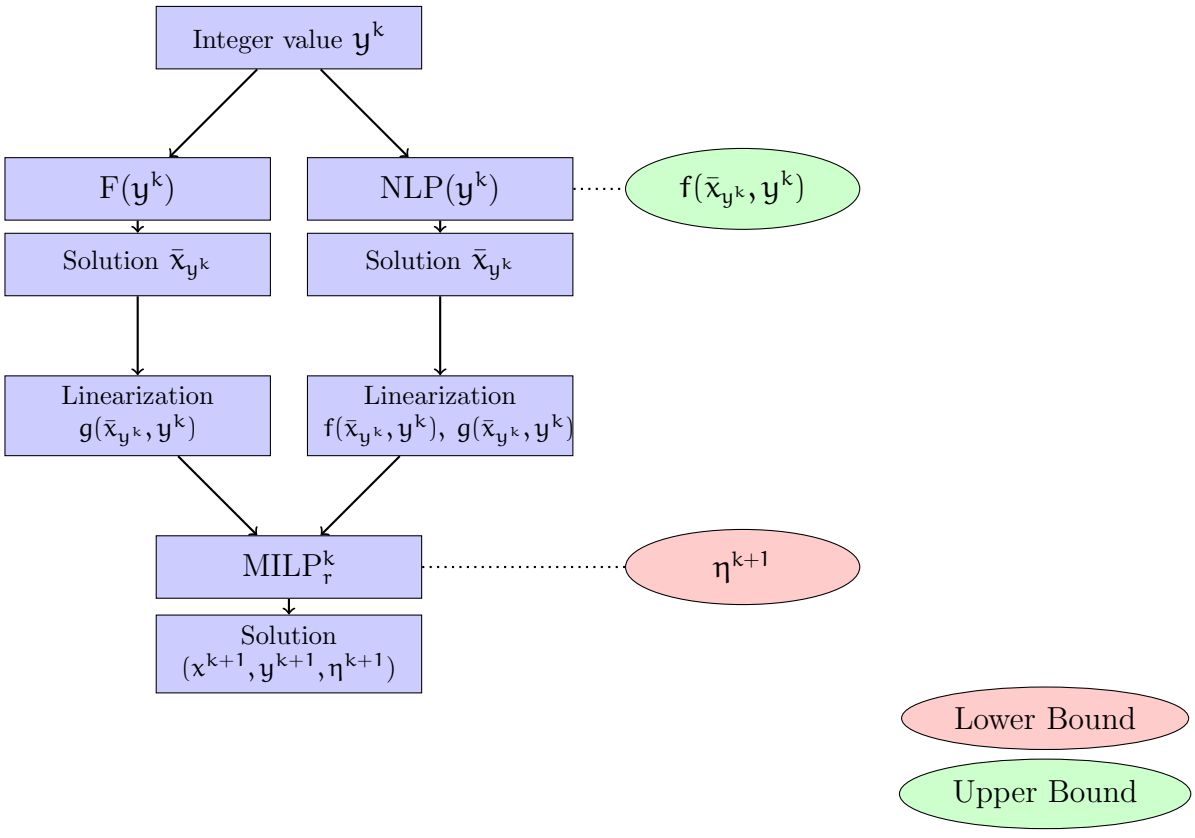

Fig. 2.5: Linear Outer Approximation

Figure 2.5 illustrates one iteration of a linear outer approximation algorithm, as e.g., proposed by Fletcher and Leyffer [50]. The iteration is started by fixing the integer

values y^k of the current iterate (x^k, y^k) and solving either NLP(y^k) or F(y^k). The solution $(\bar{x}_{y^k}, y^k)$ yields additional linearizations of the constraints and possibly the objective function. The additional linearizations are included in the master problems, which is solved to obtain the next iterate and a lower bound of MINLP (1.6). If $(\bar{x}_{y^k}, y^k)$ is the solution of NLP(y^k) it provides an upper bound.

2.6 Generalized Benders' Decomposition

The Generalized Benders' Decomposition (GBD) method was introduced by Geoffrion [57] for solving convex MINLP problems (1.6). It is a generalization of Benders' Decomposition, see Benders [22], which is applied in linear programming to solve large-scale problems. GBD is similar to linear outer approximation, described in Section 2.5. It only differs in the master problem (2.55), which is extended in each iteration by one single constraint instead of including at least the linearization of each strongly active constraint. The corresponding single inequality is called GBD cut.

If the iterate y^k in iteration k is feasible, i.e., $y^k \in Y$, where Y is given by (2.65), the continuous nonlinear programs NLP(y^k) is solved and the corresponding solution is denoted by $\bar{x}_{y^k}$, see Section 2.5. Furthermore, let $\bar{\lambda}^k \in \mathbb{R}^m$ be the Lagrangian multipliers of NLP(y^k) associated with $\bar{x}_{y^k}$. Then the following inequality is valid for MINLP (1.6), i.e., it does not cut off the optimal solution of MINLP (1.6), see Bonami et al. [28]

$$f(\bar{x}_{y^k}, y^k) + \left(\nabla_y f(\bar{x}_{y^k}, y^k) + (\bar{\lambda}^k)^T [\nabla_y g(\bar{x}_{y^k}, y^k)]\right)^T (y - y^k) \;\leq\; f(x, y). \tag{2.112}$$

If an integer iterate y^k is infeasible, i.e., $y^k \notin Y$, then one has to solve feasibility problem F(y^k), see (2.68). The corresponding solution is denoted by $\bar{x}_{y^k}$ and yields the so-called feasibility cut, which is derived instead of the GBD cut (2.112) and given by

$$(\bar{\lambda}^k)^T \left(g(\bar{x}_{y^k}, y^k) + [\nabla_y g(\bar{x}_{y^k}, y^k)]^T (y - y^k)\right) \;\geq\; 0. \tag{2.113}$$

The GBD master problem contains constraints (2.112) and (2.113) and is given by

$$x \in X,\ y \in Y,\ \eta \in \mathbb{R}: \tag{2.114}$$

$$\begin{aligned}
&\min \quad \eta, \\
&\text{s.t.} \\
&f(\bar{x}_{y^i}, y^i) + \left(\nabla_y f(\bar{x}_{y^i}, y^i) + (\bar{\lambda}^i)^T [\nabla_y g(\bar{x}_{y^i}, y^i)]\right)^T (y - y^i) \;\leq\; \eta, \quad \forall (\bar{x}_{y^i}, y^i) \in T^k, \\
&\qquad (\bar{\lambda}^j)^T \left(g(\bar{x}_{y^j}, y^j) + [\nabla_y g(\bar{x}_{y^j}, y^j)]^T (y - y^j)\right) \;\geq\; 0, \quad \forall (\bar{x}_{y^j}, y^j) \in S^k.
\end{aligned}$$

MILP (2.114) is a relaxation of MINLP (1.6) and its solution provides a lower bound on the optimal solution of MINLP (1.6), see Bonami et al. [28]. The sets T^k and S^k are defined by (2.71) and (2.72), respectively.

Since the relaxation provided by the master problem is refined in each iteration, the sequence of lower bounds is non-decreasing. The inequalities (2.112) and (2.113) are aggregations of the inequalities (2.73) and (2.74), see Abhishek, Leyffer and Linderoth [1]. As a consequence, relaxation (2.114) is weaker than the relaxation provided by the linear outer approximation master problem (2.75).

2.7 Extended Cutting Plane Method

The extended cutting plane (ECP) method, developed by Westerlund and Pettersson [108], extends Kelley's cutting plane method for solving convex NLPs, see Kelley [67], such that convex MINLPs can be solved. It can be considered as the nonlinear version of the well-known mixed-integer linear cutting plane method proposed by Gomory [59]. Moreover, there exists an extension of the ECP method for pseudo-convex functions, see Westerlund and Pörn [109].

The ECP method does not rely on the solution of any continuous nonlinear program. Instead, the convex MINLP (1.6) is solved iteratively by a series of mixed-integer linear relaxations

$$
\begin{aligned}
&x \in X,\ y \in Y,\ \eta \in \mathbb{R}: && && && (2.115)\\
&\quad \min \quad \eta, && && &&\\
&\quad \text{s.t.} \quad f(x^i, y^i) + \nabla_{x,y}\, f(x^i, y^i)^T \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} && \leq && \eta, && \forall (x^i, y^i) \in K^k,\\
&\qquad\quad g_j(x^i, y^i) + \nabla_{x,y}\, g_j(x^i, y^i)^T \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} && \geq && 0, && \forall (x^i, y^i) \in K^k,\ \ j \in \mathcal{J}(K^k).
\end{aligned}
$$

The set K^k contains the solutions (x^i, y^i) of MILP (2.115) of all previous iterations $i \leq k$ as well as the starting point (x^0, y^0). $\mathcal{J}(K^k)$ is the index set of the most violated constraints for each solution $(x^i, y^i) \in K^k$, i.e.,

$$
\mathcal{J}(K^k) := \left\{ j \in \mathcal{J} : \arg\min_{j \in \mathcal{J}} g_j(x^i, y^i) \text{ and } (x^i, y^i) \in K^k \text{ and } g_j(x^i, y^i) < -\varepsilon \right\}, \quad (2.116)
$$

where $\varepsilon \in \mathbb{R}_+$ is an appropriate feasibility tolerance. In iteration k, the solution of (2.115) yields the next iterate (x^{k+1}, y^{k+1}). Furthermore, the mixed-integer linear relaxation (2.115) is successively improved by adding the linearization of the most violated nonlinear constraint with respect to (x^k, y^k).

Instead of adding only the linearization of the most violated constraint, it is also possible to include linearizations of all violated constraints, i.e.,

$$
\tilde{\mathcal{J}}(K^k) := \left\{ j \in \mathcal{J} : g_j(x^i, y^i) < -\varepsilon \text{ with } (x^i, y^i) \in K^k \right\}. \quad (2.117)
$$

The series of solutions of problem (2.115) yields a non-decreasing sequence of lower bounds. The solution of the mixed-integer linear relaxation (2.115) corresponds to the

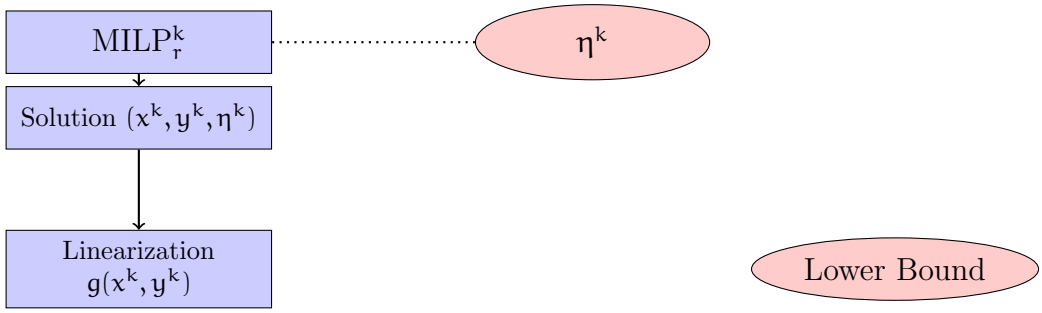

Fig. 2.6: Extended Cutting Plane Method

optimal solution of MINLP (1.6), if the maximal constraint violation is smaller than a given tolerance $\varepsilon \in \mathbb{R}_+$. Figure 2.6 illustrates an iteration of the extended cutting plane method. After solving the mixed-integer linear relaxation (2.115) in iteration k the objective value η^k provides a lower bound on the solution of MINLP (1.6). Furthermore, the relaxation is improved by adding linearizations of a selection of the constraints at (x^k, y^k).

2.8 LP/NLP-based Branch-and-Bound

LP/NLP-based branch-and-bound was developed by Quesada and Grossmann [90]. The main difference to linear outer approximation and generalized Benders' decomposition is that only one mixed-integer linear master problem (2.55) is solved. Instead of solving one MILP per iteration, the master problem is dynamically refined during the solution process by including additional linearizations of the objective function (2.73) and a selection of the constraints (2.74). The linearization can be interpreted as cutting planes and therefore the LP/NLP-based branch-and-bound method can be considered as a branch-and-cut approach for convex MINLP problems. LP/NLP-based branch-and-bound is an advanced solution method for convex MINLPs, since the solution of continuous nonlinear programs is embedded in the solution process of the MILP master problem.

The master problem is initialized by the linearizations of the objective function and the constraints at the optimal solution $(\bar{x}, \bar{y})$ of the continuous relaxation (2.47) of MINLP (1.6). MILP (2.55) is solved by a branch-and-bound method, see Section 2.4, based on its continuous linear relaxation. Whenever an integral solution with integer values $y^k \in Y$ is found during the branch-and-bound enumeration, a continuous nonlinear program is solved. Note, that k denotes the k-th integral solution found during the enumeration. If y^k is a feasible integer value, i.e., $y^k \in V$, then NLP(y^k) is solved. Otherwise, we consider F(y^k). The optimal solution of the continuous nonlinear program is denoted by $(\bar{x}_{y^k}, y^k)$, with $\bar{x}_{y^k} \in X$. If y^k is a feasible iterate, we linearize the objective function and the constraints at $(\bar{x}_{y^k}, y^k)$ to obtain the linearizations (2.73) and (2.74). If y^k is an infeasible integer value, i.e., $y \in Y \backslash V$, we generate only the

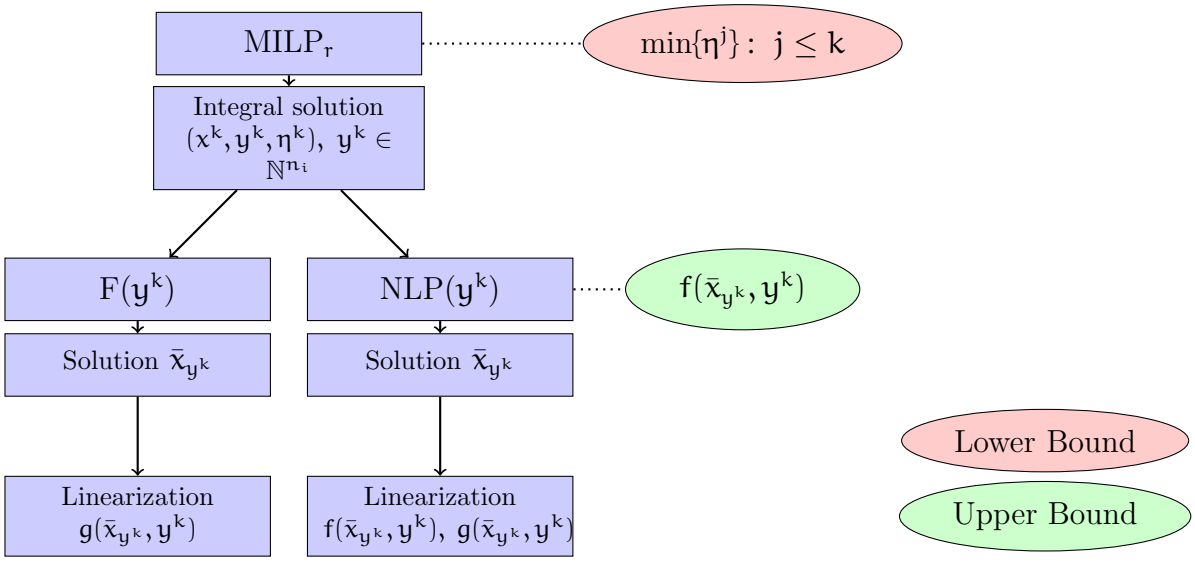

Fig. 2.7: LP/NLP-based Branch-and-Bound

linearizations (2.74) of the constraints.

The generated linearizations are included in the branch-and-bound process improving the quality of the linear relaxations of the master problem. The optimal solution of the master problem after including the additional linearizations for all feasible $y \in Y$ is the optimal solution of MINLP (1.6), see Bonami et al. [28].

Note, that the iterative improvement of the linear relaxations during branch-and-bound can be applied for linear outer approximation, generalized Benders' decomposition and the extended cutting plane method as well.

Figure 2.7 shows one iteration k of the LP/NLP-based branch-and-bound method, where an integer feasible solution (x^k, y^k, η^k) is found during the enumeration process of MILP$_r$. In this case a continuous nonlinear program, either NLP(y^k) or F(y^k), is solved and the linear relaxation defining MILP$_r$ (2.55) is refined by including the linearizations (2.74) and possibly (2.73). The lower bound on the solution of MINLP (1.6) is given by the lowest objective value of all unexplored branch-and-bound nodes of MILP$_r$. Every solution of NLP(y^k) provides an upper bound.

2.9 Integration of Branch-and-Bound and SQP

Apart from the LP/NLP-based branch-and-bound algorithm, all solution methods for convex MINLP problems presented in the previous sections decouple the solution process by considering continuous nonlinear optimization and integer optimization separately. Applying a decomposition enables the solution of problem (1.6), since efficient software for both mixed-integer linear and continuous nonlinear programming is available.

Recently, some algorithms have been proposed, reducing the computational effort for solving MINLPs by integrating mixed-integer and continuous nonlinear programming techniques. Leyffer [76] presents such a solution approach for convex MINLP problems (1.6) extending NLP-based branch-and-bound methods. The motivation is to improve the performance of the NLP-based branch-and-bound approach presented in Section 2.4 by integrating the solution process of the NLP subproblems into the branch-and-bound enumeration. This reduces the effort for solving the continuous nonlinear programs significantly, since early termination, also called early branching, is possible. Early branching was first introduced by Borchers and Mitchell [29], but Leyffer [76] improves their algorithmic ideas essentially.

As seen in Section 2.4, valid lower bounds on the optimal solution of the original problem are required at each node of the branch-and-bound tree, if bounding should be applied, i.e., the second fathoming rule stated in Corollary 2.2. If bounding cannot be applied, the efficiency of the branch-and-bound method is very poor, since a huge number of possible integer values has to be enumerated.

The goal is to reduce the computational effort for solving the continuous relaxations, see also Section 2.4, given by the following NLP in branch-and-bound iteration l.

$$
\begin{aligned}
& x \in X,\ y \in Y_{\mathbb{R}} : \\
& \quad \min \quad f(x,y) \\
& \quad \text{s.t.} \quad g_j(x,y) \ \geq \ 0, \qquad j = 1,\dots,m, \\
& \qquad\qquad y_i \ \geq \ (y_l^l)_i, \quad i \in I_L^l, \\
& \qquad\qquad y_i \ \leq \ (y_u^l)_i, \quad i \in I_U^l.
\end{aligned}
\tag{2.118}
$$

Note, that I_L^l and I_U^l, defined by (2.50) and (2.51) denote the index sets corresponding to more stringent lower and upper bounds on some integer variables, which are determined by the branch-and-bound enumeration. The idea of early branching is, to improve the performance by terminating before an optimal solution $(\bar{x}^l, \bar{y}^l)$ for NLP (2.118) is obtained. If we terminate before the optimal solution $(\bar{x}^l, \bar{y}^l)$ is determined, no valid lower bounds underestimating $\bar{f}^l := f(\bar{x}^l, \bar{y}^l)$ are available. This means, that the corresponding subtrees cannot be cut off, even if $\bar{f}^l \geq \hat{\eta}$ holds, where $\hat{\eta}$ denotes the current incumbent, i.e., the best feasible solution found so far. Note, that cutting off these subtrees is equivalent to fathoming the corresponding nodes according to the second fathoming rules stated in Corollary 2.2. One alternative way to obtain lower bounds in case of early branching is proposed by Borchers and Mitchell [29]. They suggest to evaluate the Lagrangian dual of the NLP (2.47) in iteration l for a given set of Lagrangian multipliers $\lambda \in \mathbb{R}^m$, i.e.,

$$
\begin{aligned}
& x \in X,\ y \in Y_{\mathbb{R}}^l : \\
& \quad \min \quad f(x,y) - \lambda^T g(x,y),
\end{aligned}
\tag{2.119}
$$

where $Y_{\mathbb{R}}^l$ is defined by (2.58). The solution $(\hat{x}^l, \hat{y}^l)$ of (2.119) with objective value

$$\hat{L}^l := f(\hat{x}^l, \hat{y}^l) - \lambda^T g(\hat{x}^l, \hat{y}^l) \tag{2.120}$$

provides a lower bound on the optimal solution $(\bar{x}^l, \bar{y}^l)$ of NLP (2.118), i.e., $\hat{L}^l \leq f(\bar{x}^l, \bar{y}^l)$ holds, see Leyffer [76].

Leyffer [76] presents an improved approach, where the solution of the Lagrangian dual (2.119) is unnecessary. Instead of obtaining lower bounds by solving (2.119), one can interpret the constraints of a continuous quadratic program arising during the solution process of the NLP (2.118) via a SQP method, see Section 2.2, as supporting hyperplanes. If a convex MINLP is considered, a linearization is equivalent to an outer approximation, see also Section 2.5.

If we apply a SQP method to solve NLP (2.118), a sequence of quadratic programs

$$\begin{aligned} d \in \mathbb{R}^n : \\ \min \quad & \nabla_{x,y} f(x^k, y^k)^T d + \frac{1}{2} d^T B^k d \\ \text{s.t.} \quad & g(x^k, y^k) + [\nabla_{x,y} g(x^k, y^k)]^T d \geq 0 \\ & x^k + d_x \in X \\ & y^k + d_y \in Y_{\mathbb{R}}^l \end{aligned} \tag{2.121}$$

is solved with $d = \begin{pmatrix} d_x \\ d_y \end{pmatrix}$. $B^k \in \mathbb{R}^{n \times n}$ denotes an approximation of the Hessian of the Lagrangian function and $Y_{\mathbb{R}}^l$ is defined by (2.58), see also Section 2.2. Note, that the iteration index k denotes the k-th QP subproblem while solving the l-th NLP during the branch-and-bound enumeration process. In general, the optimal objective value of QP (2.121) does not underestimate the optimal solution $f(\bar{x}^l, \bar{y}^l)$ of NLP (2.118), i.e., QP (2.121) yields no lower bound.

To obtain a lower bound of the optimal solution $f(\bar{x}^l, \bar{y}^l)$ of NLP (2.118) by the solution of the QP-subproblem, Leyffer [76] suggests to include a so-called objective cut, analogue to (2.73) and (2.76) for linear outer approximation, given by

$$f(x^k, y^k) + \nabla_{x,y} f(x^k, y^k)^T d \leq \hat{\eta} - \varepsilon. \tag{2.122}$$

$\hat{\eta} \in \mathbb{R} \cup \{\infty\}$ denotes the current incumbent. Note, that we consider the solution process of a single NLP during a branch-and-bound enumeration, which implies that

$\hat{\eta}$ does not vary. Including this objective cut in QP (2.121) yields

$$
\begin{aligned}
& d \in \mathbb{R}^n : \\
& \qquad \begin{aligned}
\min \quad & \nabla_{x,y} f(x^k, y^k)^T d + \frac{1}{2} d^T B^k d \\
\text{s.t.} \quad & g(x^k, y^k) + [\nabla_{x,y} g(x^k, y^k)]^T d \geq 0 \\
& f(x^k, y^k) + \nabla_{x,y} f(x^k, y^k)^T d \leq \hat{\eta} - \varepsilon \\
& x^k + d_x \in X \\
& y^k + d_y \in Y^l_{\mathbb{R}}.
\end{aligned}
\end{aligned}
\tag{2.123}
$$

QP (2.123) can also be interpreted as a SQP subproblem corresponding to the NLP given by

$$
\begin{aligned}
& x \in X,\ y \in Y^l_{\mathbb{R}} : \\
& \qquad \begin{aligned}
\min \quad & f(x, y) \\
\text{s.t.} \quad & g(x, y) \geq 0 \\
& f(x, y) \leq \hat{\eta} - \varepsilon.
\end{aligned}
\end{aligned}
\tag{2.124}
$$

Due to the introduction of the objective cut, bounding corresponds to the infeasibility of NLP (2.124). The following lemma stated by Leyffer [76] shows the importance of QP (2.123).

Lemma 2.7. *Let* $f(x, y)$ *and* $g(x, y)$ *be continuously differentiable functions and let* $f(x, y)$ *be convex, while* $g(x, y)$ *is concave. A sufficient condition for bounding, i.e., the second fathoming rule given by Corollary 2.2, is that QP* (2.123) *generated by a SQP method solving problem* (2.124) *is infeasible in any iteration* k.

Proof. See Leyffer [76]. □

By now we neglected the presence of a trust region ensuring global convergence. A trust region truncates the feasible region of QP (2.123). As a consequence, we have to distinguish whether infeasibility is caused by the trust region or not. If infeasibility is encountered independently of the trust region, the current subtree can be fathomed according to Lemma 2.7. If on the other hand QP (2.123) is infeasible due to the trust region constraint, a feasibility problem, similar to (2.54), has to be solved to check whether the current node can be fathomed or not, see Leyffer [76] for details.

2.10 An Extension of Yuan's Trust Region Method for Mixed-Integer Optimization

The efficient mixed-integer nonlinear SQP trust region algorithm of Exler and Schittkowski [45], whose implementation is called MISQP, is an extension of the well-known SQP methods. It is based on the trust region method proposed by Yuan [112], see Section 2.2. Analogue to continuous sequential quadratic programming methods the mixed-integer nonlinear optimization problem is solved by a sequence of mixed-integer quadratic approximations.

If we apply Yuan's trust region Algorithm 2.1 to solve the continuous relaxation of MINLP (1.1), the corresponding quadratic model is represented by

$$
\begin{aligned}
& d_x \in \mathbb{R}^{n_c},\ d_y \in \mathbb{R}^{n_i},\ \eta \in \mathbb{R}_+ : \qquad (2.125)\\
\min\quad & \nabla_{x,y} f(x^k,y^k)^T d + \tfrac{1}{2} d^T B^k d + \sigma^k \eta \\
\text{s.t.}\quad & \eta + g_j(x^k,y^k) + \nabla_{x,y} g_j(x^k,y^k)^T d \ \geq\ 0, \quad j = 1,\ldots,m_e,\\
& \eta - g_j(x^k,y^k) - \nabla_{x,y} g_j(x^k,y^k)^T d \ \geq\ 0, \quad j = 1,\ldots,m_e,\\
& \eta + g_j(x^k,y^k) + \nabla_{x,y} g_j(x^k,y^k)^T d \ \geq\ 0, \quad j = m_e+1,\ldots,m,\\
& \|d_x\|_\infty \ \leq\ \Delta_x^k,\\
& \|d_y\|_\infty \ \leq\ \Delta_y^k,\\
& x^k + d_x \ \in\ X,\\
& y^k + d_y \ \in\ Y_{\mathbb{R}},
\end{aligned}
$$

with $d := \begin{pmatrix} d_x \\ d_y \end{pmatrix}$. Since problem (2.125) is equivalent to (2.31), see Yuan [112], its solution provides a search direction in iteration k. Note, that Δ_x^k and Δ_y^k denote the continuous and integer trust region radius, see also Section 2.2.

Restricting the domain of d_y to $\mathbb{N}^{n_i}$ turns the continuous quadratic program into a

mixed-integer quadratic model of MINLP (1.1) given by problem

$$
\begin{aligned}
& d_x \in \mathbb{R}^{n_c},\ d_y \in \mathbb{N}^{n_i},\ \eta \in \mathbb{R}_+ : && && && (2.126)\\
\min\quad & \nabla_{x,y} f(x^k, y^k)^T d + \tfrac{1}{2} d^T B^k d + \sigma^k \eta && && &&\\
\text{s.t.}\quad & \eta + g_j(x^k, y^k) + \nabla_{x,y} g_j(x^k, y^k)^T d && \geq && 0, && j = 1, \ldots, m_e,\\
& \eta - g_j(x^k, y^k) - \nabla_{x,y} g_j(x^k, y^k)^T d && \geq && 0, && j = 1, \ldots, m_e,\\
& \eta + g_j(x^k, y^k) + \nabla_{x,y} g_j(x^k, y^k)^T d && \geq && 0, && j = m_e + 1, \ldots, m,\\
& \|d_x\|_\infty && \leq && \Delta_x^k, &&\\
& \|d_y\|_\infty && \leq && \Delta_y^k, &&\\
& x^k + d_x && \in && X, &&\\
& y^k + d_y && \in && Y. &&
\end{aligned}
$$

Applying Yuan's trust region method, were QP subproblems are replaced by MIQP (2.126) ensures, that each iterate (x^k, y^k) satisfies the integrality condition, i.e., $y^k \in \mathbb{N}^{n_i}$. Problem (2.126) can be solved by any mixed-integer quadratic solver, such as MIQL of Lehmann and Schittkowski [71]. Extensive tests on academic and real-world problems show, that the algorithm is very efficient in terms of the number of function evaluations. Furthermore, the algorithm is only based on local approximations, i.e., the number of linearizations is not successively increasing and linearizations only depend on the current iteration point (x^k, y^k). As a consequence, it performs very well for non-convex problems.

Despite of the encouraging results, no convergence proof for this mixed-integer nonlinear SQP trust region algorithm was found yet. Furthermore, extensive test with the current implementation called MISQP showed, that in some rare cases the global optimal solution of a convex test-case was not found. In future work, we will focus on a convergence proof for an algorithm, that is based on the ideas implemented in MISQP but where slight modifications are incorporated due to the existence of convex problems, where MISQP fails. Since only local approximations are contained MIQP (2.126) provides no valid lower bounds. Furthermore, the adjustment of some important parameters such as the trust region radius Δ_y^k in (2.126) is critical, especially for binary variables.

2.11 Convex Mixed-Integer Quadratic Programming

One main focus of this thesis is the development of efficient solution techniques and appropriate theory for strictly convex mixed-integer quadratic optimization problems.

Efficient solvers for convex MIQP problems are needed in order to solve larger MINLP problems based on the successive solution of convex MIQP subproblems. Some corresponding MINLP solvers are MISQP, introduced in Section 2.10, and MIQPSOA, proposed in Chapter 3.

To obtain a fast and robust solver for convex MIQP problems, we want to focus on branch-and-cut methods, since the concept of combining the branch-and-bound enumeration with the generation of cutting planes led to powerful state-of-the-art mixed-integer linear solution methods. Despite the huge progress of MILP solvers within the last twenty years, the solution of MIQPs by the QP-based branch-and-bound approach does not profit from these developments. The main difference between NLP-based branch-and-bound methods for solving MINLPs and QP-based branch-and-bound algorithms for solving MIQPs, is the reduced effort needed to solve a QP instead of a NLP. Recent developments for MIQP- and mixed-integer quadratically constrained programming (MIQCP)-solvers concentrate on competitive approaches relying on linear relaxations, see Section 2.7 and Section 2.8 for related nonlinear methods and Berthold, Heinz and Vigerske [24] for a comparative study.

In addition to cutting planes, state-of-the-art MILP solvers contain lots of additional components improving the performance of the branch-and-bound enumeration. The integration of these techniques turned mixed-integer linear solvers into very powerful algorithms. Their tremendous improvement is based on three major components, as presented by Bixby for the solver CPLEX of IBM/ILOG [41], see Table 1.1.

Apart from a large variety of cut generators, powerful continuous linear solvers possess excellent warmstart features. They are a key component of an efficient branch-and-cut solver. A warmstart allows the reduction of the computational effort for solving an optimization problem by exploiting information of previous runs for similar problems. Furthermore, advanced presolve techniques reduce the problem complexity prior to the solution process. Apart from these techniques other components helped to increase the power of state-of-the-art mixed-integer linear solvers, e.g., heuristics, see Berthold [23].

Since the feasible region of both MIQP and MILP is a polyhedron, all mixed-integer linear techniques, that do not rely on the objective function, might be applicable. Therefore, all presolving procedures, that do not take the objective function into account can be directly integrated in a MIQP solver. As cutting planes only depend on the feasible region, they can in principle be applied for quadratic programs as well. Their application for MIQPs involves some difficulties, which will be explained in detail in Chapter 4 and Chapter 5. In this section we describe a continuous quadratic solver focusing on warmstarts, which are also beneficial for continuous quadratic programs, see Chapter 6.

We consider the following strictly convex mixed-integer quadratic optimization problem, possessing a strictly convex quadratic objective function as well as linear equality

and inequality constraints.

$$
\begin{aligned}
& x \in X,\ y \in Y: \\
& \qquad \begin{array}{lrcl}
\min & \frac{1}{2}\left(x^T, y^T\right) B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix} & & \\
\text{s.t.} & A_E \begin{pmatrix} x \\ y \end{pmatrix} & = & b_E, \\
& A_I \begin{pmatrix} x \\ y \end{pmatrix} & \geq & b_I.
\end{array}
\end{aligned}
\tag{2.127}
$$

x and y denote the vectors of the continuous and integer variables, respectively, while $B \in \mathbb{R}^{n \times n}$ is a positive definite matrix and $c \in \mathbb{R}^n$ holds. X and Y are defined by the upper and lower bounds on both the continuous and the integer variables, see (1.2). n_c denotes the number of continuous variables and n_i is the number of integer variables. The total number of variables is denoted by n, i.e., $n := n_i + n_c$. Equality constraints are denoted by $A_E \in \mathbb{R}^{m_e \times n}$ and $b_E \in \mathbb{R}^{m_e}$, while inequality constraints are given by $A_I \in \mathbb{R}^{m_i \times n}$ and $b_I \in \mathbb{R}^{m_i}$. Therefore m_e denotes the number of equality constraints, while m_i is the number of inequality constraints.

To be consistent with the notation used in the subsequent chapters, a variable transformation is carried out, such that

$$
\begin{pmatrix} x_l \\ y_l \end{pmatrix} = 0 \tag{2.128}
$$

holds, where x_l und y_l are lower bounds on the continuous and integral variables, see (1.2). Furthermore, problem (2.127) is reformulated yielding

$$
\begin{aligned}
& x \in \mathbb{R}^{n_c},\ y \in \mathbb{R}^{n_i}: \\
& \qquad \begin{array}{lrcl}
\min & \frac{1}{2}\left(x^T, y^T\right) B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix} & & \\
\text{s.t.} & \hat{A} \begin{pmatrix} x \\ y \end{pmatrix} & \geq & \hat{b}.
\end{array}
\end{aligned}
\tag{2.129}
$$

The number of constraints in (2.129) is denoted by $\hat{m}$, with $\hat{m} := m_i + 2m_e + 2n$. The matrix $\hat{A} \in \mathbb{R}^{\hat{m} \times n}$ and the vector $\hat{b} \in \mathbb{R}^{\hat{m}}$ are therefore given by

$$
\hat{A} := \begin{bmatrix} A_I \\ A_E \\ -A_E \\ -I \\ I \end{bmatrix}, \qquad \hat{b} := \begin{bmatrix} b_I \\ b_E \\ -b_E \\ \begin{pmatrix} x_u \\ y_u \end{pmatrix} \\ 0 \end{bmatrix}, \tag{2.130}
$$

where x_u and y_u are modified due to the variable transformation ensuring (2.128). In this formulation $\hat{m} \geq n$ holds, due to the existence of upper and lower bounds

for every variable. We denote by $\hat{\mathbb{J}}$ the index set of all constraints of MIQP (2.129), while $\mathbb{J} := \mathbb{J}_= \cup \mathbb{J}_>$ is the index set of the constraint of MIQP formulation (2.127) without box-constraints, where equality constraints are indexed by $\mathbb{J}_=$ and inequality constraints are indexed by $\mathbb{J}_>$.

The objective function of (2.129) is abbreviated by

$$f_{qp}(x, y) := \frac{1}{2}(x^T, y^T) B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix}. \tag{2.131}$$

Furthermore, the integer variables $y \in \mathbb{N}^{n_i}$ are indexed by the set $I \subset \{1, \ldots, n\}$ and the indices of the continuous variables are given by the set $J = \{1, \ldots, n\} \backslash I$.

First, we briefly review a solution method for strictly convex, continuous, quadratic optimization problems, that was proposed by Goldfarb and Idnani [58]. The algorithm is a dual approach for solving problem QP^l given by

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{R}^{n_i} : \\ & \qquad \min \quad \frac{1}{2}(x^T, y^T)\, B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix} \\ & \qquad \text{s.t.} \quad \hat{A}^l \begin{pmatrix} x \\ y \end{pmatrix} \ \geq \ \hat{b}^l, \end{aligned} \tag{2.132}$$

where l denotes the iteration index of a branch-and-bound method applied to solve MIQP (2.129). As a consequence, the constraints given by $(\hat{A}^l, \hat{b}^l)$ include additional box-constraints tightening the domain of the relaxed integer variables y, i.e., $y \in Y^l_{\mathbb{R}}$ given by (2.50) and (2.51). The corresponding index set of the constraints is denoted by $\hat{\mathbb{J}}^l$

For $l = 0$, QP^l (2.132) is the continuous relaxation of MIQP (2.129), where the integrality condition for the integer variables $y \in \mathbb{N}^{n_i}$ is relaxed, i.e., $y \in \mathbb{R}^{n_i}$.

Usually the method of Goldfarb and Idnani [58] is applied for solving subproblems arising during the solution of nonlinear programs using a SQP method, see Section 2.2. In this case, the matrix B can be associated with a Quasi-Newton approximation introduced in Definition 2.7 and the vector $c \in \mathbb{R}^n$ corresponds to the gradient of the nonlinear objective function $f(x, y)$. Moreover, the constraints are linearizations of the nonlinear constraints $g(x, y)$, see NLP (2.1).

We consider a fixed branch-and-bound iteration first, e.g., the continuous relaxation of MIQP (2.127) for $l = 0$ and therefore remove the corresponding index to ease the readability. For the remainder of this section k denotes the iteration index of the method of Goldfarb and Idnani.

The dual method of Goldfarb and Idnani [58] is an iterative active set method. In each iteration k the active set consists of a linear independent subset of the constraints of QP (2.132). All constraints belonging to the current active set are satisfied as equalities in the corresponding iteration. The active set of linear independent constraints satisfied with as equalities in iteration k is denoted by $\mathbb{A}^k \subset \hat{\mathbb{J}}$.

The remaining constraints not contained in the active set $\mathbb{A}^k$ are either satisfied or violated. Removing all constraints j, with $j \notin \mathbb{A}^k$, we obtain a relaxation of QP (2.132). This continuous quadratic program is denoted by $\mathrm{QP}(\mathbb{A}^k)$ and is given by

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{R}^{n_i}: \\ & \quad \min \quad \frac{1}{2}(x^T, y^T)\, B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix} \\ & \quad \text{s.t.} \quad \hat{a}_j^T \begin{pmatrix} x \\ y \end{pmatrix} \ \geq \ \hat{b}_j, \quad j \in \mathbb{A}^k, \end{aligned} \tag{2.133}$$

where $\hat{a}_j \in \mathbb{R}^n$ is a row of $\hat{A}$ and $\hat{b}_j$ is an entry in $\hat{b}$.

Since the method is a dual approach, violated constraints always exist as long as the optimal solution is not obtained. For each constraint not contained in the current active set $\mathbb{A}^k$ the slack value introduced below at some iterate $(x^k, y^k) \in \mathbb{R}^n$ determines, whether or not the constraint is violated:

$$s_j^k \ := \quad \hat{a}_j^T \begin{pmatrix} x^k \\ y^k \end{pmatrix} - \hat{b}_j, \quad j \in \hat{\mathbb{J}}. \tag{2.134}$$

The algorithm is based on so-called solution pairs, defined as follows.

Definition 2.16. *$((x^k, y^k), \mathbb{A}^k)$ with $(x^k, y^k) \in \mathbb{R}^n$ and $\mathbb{A}^k \subset \hat{\mathbb{J}}$ is a solution pair of QP($\mathbb{A}^k$) (2.133), abbreviated by S-pair, if (x^k, y^k) is the optimal solution of QP($\mathbb{A}^k$) and the constraints contained in $\mathbb{A}^k$ are linear independent and satisfied as equalities, i.e. active.*

Note, that (x^k, y^k) is the optimal solution of $\mathrm{QP}(\{j \in \hat{\mathbb{J}} : s_j^k \geq 0\})$, where s_j^k is defined in (2.134), if $((x^k, y^k), \mathbb{A}^k)$ is an S-pair.

The algorithm of Goldfarb and Idnani can be briefly described by the following iteration sequence.

Algorithm 2.3. *1. Let $((x^k, y^k), \mathbb{A}^k)$ be a S-pair of QP($\mathbb{A}^k$).*

2. Iterate until all constraints of QP (2.132) are satisfied:

(a) Choose a constraint of QP (2.132), which is violated by the solution (x^k, y^k) of QP($\mathbb{A}^k$) and denote the corresponding index by j^*.

(b) Determine next S-pair of QP($\mathbb{A}^{k+1}$), if possible:

If *$QP\big(\mathbb{A}^k \cup \{j^*\}\big)$ is infeasible,* **then stop**, *since QP (2.132) is infeasible.*

Else *determine new S-pair $((x^{k+1}, y^{k+1}), \mathbb{A}^{k+1})$ of QP($\mathbb{A}^{k+1}$) with $\mathbb{A}^{k+1} := \hat{\mathbb{A}}^k \cup \{j^*\}$, where $\hat{\mathbb{A}}^k$ is as suitable subset of $\mathbb{A}^k$, i.e. $\hat{\mathbb{A}}^k \subset \mathbb{A}^k$, and $f_{qp}(x^{k+1}, y^{k+1}) > f_{qp}(x^k, y^k)$ holds. Set $(x^k, y^k, \mathbb{A}^k) := ((x^{k+1}, y^{k+1}), \mathbb{A}^{k+1})$.*

The initial S-pair of $QP(\mathbb{A}^0)$ is $((x^0, y^0), \emptyset)$, where (x^0, y^0) denotes the unrestricted minimum of QP (2.132), given by

$$\nabla_{x,y} f_{qp}(x, y) = 0, \tag{2.135}$$

i.e., (x^0, y^0) is obtained by

$$(x^0, y^0) = -B^{-1}c. \tag{2.136}$$

Subsuming the basic Algorithm 2.3, we know that in each iteration, the current iterate (x^k, y^k) solves $QP(\mathbb{A}^k)$, i.e., $((x^k, y^k), \mathbb{A}^k)$ is a solution pair. If (x^k, y^k) satisfies all constraints of QP (2.132), then (x^k, y^k) is the optimal solution of QP (2.132). Otherwise a new solution pair is obtained after a finite number of internal steps, see Goldfarb and Idnani [58], Werner [107] or Lehmann [70] for algorithmic details. If QP (2.132) is not detected to be infeasible, we know that the sequence of solution pairs cannot cycle, since

$$f_{qp}^{k+1}(x^{k+1}, y^{k+1}) > f_{qp}^{k}(x^k, y^k) \tag{2.137}$$

holds. Furthermore, the number of solution pairs is finite and therefore the optimal solution of QP (2.132) is obtained or infeasibility is detected.

The briefly introduced method of Goldfarb and Idnani [58] can serve as one of the core components of an efficient branch-and-cut method for MIQPs, since it is a reliable and fast algorithm for solving dense, continuous QP problems up to medium size. These continuous QP problems naturally arise in a branch-and-bound algorithm, which is based on quadratic relaxations. Since we want to develop a MIQP solver inspired by state-of-the-art MILP solution techniques, we extend the original algorithm based on an efficient implementation of Schittkowski [94]. One major feature of the extended method is the ability to perform warmstarts.

We refer to warmstarts, whenever we can reduce the computational effort, e.g., calculation time, for solving an optimization problem by exploiting information obtained from the solution process of a related problem. During the branch-and-bound enumeration a large number of similar subproblems arises. We want to exploit the corresponding close relationship by performing warmstarts. There are two specific kinds of relationships, where warmstarts naturally reduce the effort for solving related QPs.

Note, that we consider in the remainder of this section different continuous QPs arising during a branch-and-bound enumeration applied to MIQP (2.127). The optimal solution of the $l-th$ QP is denoted by $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l) \in X \times Y_{\mathbb{R}}^l \times \mathbb{R}^{\hat{m}^l}$. The domain of the relaxed integer variables $Y_{\mathbb{R}}^l$, defined by (2.58), contains bound changes given by I_L^l and I_U^l, which are performed by the branch-and-bound method, see (2.50) and (2.51). These box-constraints tightening the original feasible domain $Y_{\mathbb{R}}$ of the relaxed integer variables are contained in the constraints $(\hat{A}^l, \hat{b}^l)$. $\hat{m}^l$ denotes the number of constraints indexed by $\hat{\mathbb{J}}^l$ and $\bar{\mathbb{A}}^l$ is the set of linear independent and active constraints at $(\bar{x}^l, \bar{y}^l)$. Furthermore, $\mathbb{F}^l$ denotes the feasible region of QP^l.

In the subsequent definition we introduce a dual warmstart.

Definition 2.17. *Let* $((\bar{x}^l, \bar{y}^l), \bar{\mathbb{A}}^l)$ *be the solution pair solving* QP^l (2.132)*, with* $(\bar{x}^l, \bar{y}^l) \in X \times Y^l_{\mathbb{R}}$.

Then $((\bar{x}^l, \bar{y}^l), \bar{\mathbb{A}}^l)$ *allows a dual warmstart for* QP^{l+1}*, if* $((\bar{x}^l, \bar{y}^l), \bar{\mathbb{A}}^l)$ *is a solution pair of some relaxation of* QP^{l+1} *according to Definition 2.16.*

A warmstart can be performed, if the solution of the previous subproblem is optimal for a subset of the constraints defining the successive, continuous, quadratic subproblem. If we assume that the the solution $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ of QP^l is not optimal for QP^{l+1}, some constraints of QP^{l+1} are still violated at $(\bar{x}^l, \bar{y}^l)$. If many of the constraints of the successive QP are already satisfied, the solution process can be speeded up by exploiting this knowledge.

The following relationship between two successive quadratic programs QP^l and QP^{l+1} always allows a dual warmstart.

Corollary 2.6. *Let* $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ *be the solution of* QP^l (2.132)*, with* $(\bar{x}^l, \bar{y}^l) \in X \times Y^l_{\mathbb{R}}$ *and* $\bar{\lambda}^l \in \mathbb{R}^{\hat{m}^l}$*. Consider* QP^{l+1} *with* $\hat{\mathbb{J}}^l \subset \hat{\mathbb{J}}^{l+1}$*, then* $\mathbb{F}^{l+1} \subset \mathbb{F}^l$ *holds. As a consequence,* $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ *is optimal for a subset of the constraints of* QP^{l+1} *and a dual warmstart can be performed.*

We define a primal warmstart in the following way.

Definition 2.18. *Let* $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ *be the solution of* QP^l (2.132)*, with* $(\bar{x}^l, \bar{y}^l) \in X \times Y^l_{\mathbb{R}}$ *and* $\bar{\lambda}^l \in \mathbb{R}^{\hat{m}^l}$*.* $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ *allows a primal warmstart for* QP^{l+1}*, if* $(\bar{x}^l, \bar{y}^l)$ *is primal feasible for* QP^{l+1}*, i.e.,* $(\bar{x}^l, \bar{y}^l) \in \mathbb{F}^{l+1}$.

A primal feasible solution of a previous subproblem QP^l allows an efficient warmstart for solving QP^{l+1}. In this situation, a feasible point is known and one has to regain optimality. Since finding a feasible point for a quadratic program determines a significant amount of the total solution effort of a primal method, see Goldfarb and Idnani [58], a warmstart is beneficial.

In the following case a primal feasible solution is always available.

Corollary 2.7. *Let* $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ *be the solution of* QP^l (2.132)*, with* $(\bar{x}^l, \bar{y}^l) \in X \times Y^l_{\mathbb{R}}$ *and* $\bar{\lambda}^l \in \mathbb{R}^{\hat{m}^l}$*. Consider* QP^{l+1} *with* $\hat{\mathbb{J}}^{l+1} \subset \hat{\mathbb{J}}^l$*, then* $\mathbb{F}^l \subset \mathbb{F}^{l+1}$ *holds. In this situation,* $(\bar{x}^l, \bar{y}^l)$ *is primal feasible for* QP^{l+1} *and a primal warmstart can be performed.*

Since the algorithm of Goldfarb and Idnani [58], presented in the previous section, is a dual method, it is very well suited to perform dual warmstarts. Storing the information obtained during the solution of the previous subproblem, a dual warmstart can be performed with very little effort. If the solution of QP^l is a solution pair of some relaxation of QP^{l+1} according to Definition 2.16, the solution process can simply be continued, due to Definition 2.17. The aim is to satisfy all additional, violated constraints. Especially, if only very few additional constraints are to be satisfied, a dual warmstart is very efficient, see Chapter 6.

A primal warmstart cannot be performed by the method of Goldfarb and Idnani [58] in a straightforward way. Nevertheless, one can exploit the factored QR decomposition of the previous run, to regain optimality with a primal approach.

Within a branch-and-cut solver, there are many situations, where dual or primal warmstarts can be performed. During the branch-and-bound enumeration, the node selection strategy determining the subsequent node in the tree to continue, heavily influences the ability to perform warmstarts. Using depth-first-search, i.e., selecting a child node whenever possible, allows almost always warmstarts. In contrast, the node selection rule best-first-search yields only few situations, in which a warmstart can be performed, since subsequent nodes usually lie in different parts of the search-tree.

Furthermore, the potential reduction of the computational effort by a dual warmstart is also significant, if cutting planes are found and need to be incorporated into the current problem formulation.

Warmstarts lead to a significant reduction of the computation time, if the successive subproblems are closely related, see Chapter 6. The following situations are often encountered in a branch-and-cut solver. We restrict our considerations concerning the branch-and-bound enumeration on the standard case of binary branching on an integer variable $y_i \in Y_{\mathbb{R}}^l$ with fractional value $\bar{y}_i^l \in \mathbb{R}\backslash\mathbb{N}$ at the optimal solution $(\bar{x}^l, \bar{y}^l)$ of the current node corresponding to QP^l.

1. The new node is a child of the current one.

   Considering iteration l of the branch-and-bound enumeration process, then $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ is the solution of QP^l (2.132), with $(\bar{x}^l, \bar{y}^l) \in X \times Y_{\mathbb{R}}^l$ and $\bar{\lambda}^l \in \mathbb{R}^{\hat{m}^l}$. Furthermore, let the subsequent node $l+1$, corresponding to QP^{l+1}, be a child of node l where branching is performed on variable $y_i \in Y_{\mathbb{R}}^l$. Then $\mathbb{J}^{l+1} = \mathbb{J}^l$ and with $Y_{\mathbb{R}}^{l+1} \subset Y_{\mathbb{R}}^l$, since either $I_L^{l+1} = I_L^l$ and $I_U^{l+1} = I_U^l \cup \{i\}$ holds, if node $l+1$ has to satisfy branching condition $y_i \geq \lceil \bar{y}_i^l \rceil$ or $I_L^{l+1} = I_L^l \cup \{i\}$ and $I_U^{l+1} = I_U^l$ holds, if node $l+1$ has to satisfy branching condition $y_i \leq \lfloor \bar{y}_i^l \rfloor$ respectively. Therefore, $\mathbb{F}^{l+1} \subset \mathbb{F}^l$ holds and $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ is optimal for a relaxation of QP^{l+1} and we can perform a dual warmstart according to Corollary 2.6.

   Since node $l+1$ is a child of node l and we perform a binary branching on variable $y_i \in Y_{\mathbb{R}}^l$, the new quadratic program and the previous one are identical apart from one additional box constraint subject to y_i. Since the dual method of Goldfarb and Idnani [58] successively adds constraints to the active set, we need in most cases only few additional iterations to get a new optimal solution of QP^{l+1}. Therefore, a significant amount of calculation time is saved compared to solving the complete quadratic program QP^{l+1} from scratch.

2. The new node has the same parent node as the current one.

   Considering iteration l of the branch-and-bound enumeration process, then $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ is the solution of QP^l (2.132), with $(\bar{x}^l, \bar{y}^l) \in X \times Y_{\mathbb{R}}^l$ and $\bar{\lambda}^l \in \mathbb{R}^{\hat{m}^l}$. Furthermore, let the subsequent node $l+1$, corresponding to QP^{l+1}, possess

the same parent node with index $\hat{l} < l$ as the current node l and assume, that branching was performed on variable $y_i \in Y_{\mathbb{R}}^{\hat{l}}$. Then $\mathbb{J}^{l+1} = \mathbb{J}^l$ and with $Y_{\mathbb{R}}^{l+1} \not\subset Y_{\mathbb{R}}^l$, since either $I_L^{l+1} = I_L^l \backslash \{i\}$ and $I_U^{l+1} = I_U^l \cup \{i\}$ holds, if node $l+1$ has to satisfy branching condition $y_i \geq \lceil \bar{y}_i^{\hat{l}} \rceil$ or $I_L^{l+1} = I_L^l \cup \{i\}$ and $I_U^{l+1} = I_U^l \backslash \{i\}$ holds, if node $l+1$ has to satisfy branching condition $y_i \leq \lfloor \bar{y}_i^{\hat{l}} \rfloor$ respectively.

As a consequence, $\mathbb{F}^{l+1} \not\subset \mathbb{F}^l$ holds and $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ is not optimal for a relaxation of QP^{l+1} and we can neither perform a dual nor a primal warmstart. Nevertheless, computational effort can be significantly reduced, since both problems differ in only one box constraint for an integer variable. QP^{l+1} can be solved by combining a dual and a primal warmstart, first regaining optimality for $QP^{\hat{l}}$ by a primal warmstart and then satisfying the violated constraint induced by branching via a dual warmstart.

3. The new node is the child of a node, which has the same parent node as the current one (*nephew*).

   Considering iteration l of the branch-and-bound enumeration process, then $(\bar{x}^l, \bar{y}^l, \bar{\lambda}^l)$ is the solution of QP^l (2.132), with $(\bar{x}^l, \bar{y}^l) \in X \times Y_{\mathbb{R}}^l$ and $\bar{\lambda}^l \in \mathbb{R}^{\hat{m}^l}$. Furthermore, let the subsequent node $l+1$, corresponding to QP^{l+1}, be the nephew of the current node l, i.e., node $l+1$ is a grandchild of the parent node, with index $\hat{l} < l$, of the current node l. Furthermore, let the parent node of node $l+1$, i.e., the sibling of the current node l, be indexed by $\bar{l}$ with $\hat{l} < \bar{l} < l$. Assume, that at node $\hat{l}$ branching was performed on variable $y_i \in Y_{\mathbb{R}}^{\hat{l}}$ and at node $\bar{l}$ branching was performed on variable $y_j \in Y_{\mathbb{R}}^{\bar{l}}$. Then $\mathbb{J}^{l+1} = \mathbb{J}^l$ and $Y_{\mathbb{R}}^{l+1} \not\subset Y_{\mathbb{R}}^l$, since branching is executed such that one of the subsequent four possibilities hold.

   (a) $I_L^{l+1} = I_L^l \backslash \{i\} \cup \{j\}$ and $I_U^{l+1} = I_U^l \cup \{i\}$ holds, if node $l+1$ has to satisfy branching conditions $y_i \geq \lceil \bar{y}_i^{\hat{l}} \rceil$ and $y_j \leq \lfloor \bar{y}_j^{\bar{l}} \rfloor$.

   (b) $I_L^{l+1} = I_L^l \backslash \{i\}$ and $I_U^{l+1} = I_U^l \cup \{i\} \cup \{j\}$ holds, if node $l+1$ has to satisfy branching conditions $y_i \geq \lceil \bar{y}_i^{\hat{l}} \rceil$ and $y_j \geq \lceil \bar{y}_j^{\bar{l}} \rceil$.

   (c) $I_L^{l+1} = I_L^l \cup \{i\} \cup \{j\}$ and $I_U^{l+1} = I_U^l \backslash \{i\}$ holds, if node $l+1$ has to satisfy branching conditions $y_i \leq \lfloor \bar{y}_i^{\hat{l}} \rfloor$ and $y_j \leq \lfloor \bar{y}_j^{\bar{l}} \rfloor$.

   (d) $I_L^{l+1} = I_L^l \cup \{i\}$ and $I_U^{l+1} = I_U^l \backslash \{i\} \cup \{j\}$ holds, if node $l+1$ has to satisfy branching conditions $y_i \geq \lfloor \bar{y}_i^{\hat{l}} \rfloor$ and $y_j \geq \lceil \bar{y}_j^{\bar{l}} \rceil$.

   As a consequence, $\mathbb{F}^{l+1} \not\subset \mathbb{F}^l$ holds and (x^l, y^l, λ^l) is not optimal for a relaxation of QP^{l+1} and we can neither perform a dual nor a primal warmstart. Nevertheless computational effort can be significantly reduced, since both problems differ only slightly. QP^{l+1} can be solved by combining a dual and a primal warmstart, first regaining optimality for $QP^{\hat{l}}$ by the primal warmstart and then satisfying the violated constraint induced by branching on $y_i \in Y_{\mathbb{R}}^{\hat{l}}$ and on $y_j \in Y^{\bar{l}}$ via a dual warmstart.

4. Integration of cutting planes.

   Considering cut generation round r, see Chapter 4 for details, then $(\bar{x}^{0,r}, \bar{y}^{0,r}, \bar{\lambda}^{0,r})$ is the solution of $QP^{0,r}$. In cut generation round 0, $QP^{0,0}$ corresponds to the continuous relaxation QP^0 (2.132), i.e., $(\bar{x}^{0,r}, \bar{y}^{0,r}) \in X \times Y^0_{\mathbb{R}}$ with $I^0_L = I^0_U = \emptyset$. In each round r the number of linear constraints $\hat{m}^{0,r}$ is increased by the number $\hat{m}_r \in \mathbb{N}$ of generated cutting planes. Therefore, $\bar{\lambda}^{0,r} \in \mathbb{R}^{\hat{m}^{0,r}}$ holds and the subsequent subproblem $QP^{0,r+1}$ is obtained from $QP^{0,r}$ by integrating $\hat{m}_r > 0$ linear cutting planes. As a consequence, the number of linear constraints is increased from $\hat{m}^{0,r}$ to $\hat{m}^{0,r+1} := \hat{m}^{0,r} + \hat{m}_r$. Due to $\hat{\mathbb{J}}^{0,r} \subset \hat{\mathbb{J}}^{0,r+1}$, $\mathbb{F}^{0,r+1} \subset \mathbb{F}^{0,r}$ holds and $(\bar{x}^{0,r}, \bar{y}^{0,r}, \bar{\lambda}^{0,r})$ is optimal for a relaxation of $QP^{0,r+1}$ and we can perform a dual warmstart according to Corollary 2.6.

If a primal warmstart is only needed to regain optimality for closely related problems, i.e., the next node is a sibling or a nephew, see above, one can avoid a primal warmstart. Since we know in advance, that we might want to perform a primal warmstart to regain optimality for the parent node $\hat{l}$ of the current node l, we can store the optimal solution $(\bar{x}^{\hat{l}}, \bar{y}^{\hat{l}}, \bar{\lambda}^{\hat{l}})$ and the active set $\bar{\mathbb{A}}^{\hat{l}}$ of $QP^{\hat{l}}$. Note, that $((\bar{x}^{\hat{l}}, \bar{y}^{\hat{l}}), \bar{\mathbb{A}}^{\hat{l}})$ is a solution pair of some relaxation of problem QP^{l+1}. Now we can compare the active sets $\bar{\mathbb{A}}^{\hat{l}}$ and $\bar{\mathbb{A}}^{l}$ and update the QR decomposition by adding missing constraints and removing surplus constraints, by exploiting the updating schemes of the method of Goldfarb and Idnani. Afterwards the QR decomposition corresponds to $\bar{\mathbb{A}}^{\hat{l}}$, but the values of primal and dual variables differ, such that $(\tilde{x}, \tilde{y}, \tilde{\lambda})$ obtained by this operation, is not the optimal solution of $QP^{\hat{l}}$. Replacing the current non-optimal values $(\tilde{x}, \tilde{y}, \tilde{\lambda})$ by $(\bar{x}^{\hat{l}}, \bar{y}^{\hat{l}}, \bar{\lambda}^{\hat{l}})$, which were stored, optimality of $QP^{\hat{l}}$ is regained.

It is possible, that warmstarts lead to numerical instabilities based on round-off errors. To avoid this situation the number of successive warmstarts should be limited. In case of numerical errors, the problem can be automatically resolved.

3. A NEW MIQP-BASED MINLP SOLUTION METHOD

In this chapter we propose a new algorithm for solving convex MINLPs. It is inspired by both the linear outer approximation approach and MISQP, see Section 2.5 and Section 2.10. The aim is to combine the advantages of both methods. First of all the reliability of linear outer approximation for non-convex problems is to be improved. In addition, the number of function evaluations, that are required during the solution process, should be decreased significantly, as they are an important performance criterion for industrial applications based on expensive simulations. Furthermore, we want to guarantee convergence properties for convex MINLPs. In Section 3.1, we propose an extension of a linear outer approximation method incorporating integer search steps obtained by the solution of strictly convex MIQP problems. In Section 3.2, we prove convergence of the new outer approximation algorithm for convex MINLPs. Finally, we motivate future research, yielding an algorithm, which is only based on the successive solution of MIQP problems and which ensures convergence properties for convex MINLP problems. Furthermore, we address some implementation aspects.

3.1 MIQP-Supported Linear Outer Approximation

In this section we propose a new outer approximation algorithm, which incorporates mixed-integer search steps obtained from the solution of strictly convex MIQP problems. The algorithm is designed such that convergence properties can be established under realistic conditions for the convex MINLP problem given by

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{N}^{n_i}: \\ & \quad \min \quad f(x,y) \\ & \quad \text{s.t.} \quad g_j(x,y) \ \geq \ 0, \quad j \in \mathbb{J}. \end{aligned} \tag{3.1}$$

Within this chapter the box-constraints are included in the constraints $g_j,\ j \in \mathbb{J}$. Therefore we denote the number of the original nonlinear constraints by $\tilde{m}$ for the moment and extend these constraints by n_c upper and n_c lower bounds on the con-

tinuous variables and n_i upper and n_i lower bounds on the integer variables, i.e.,

$$\begin{aligned} g_{\tilde{m}+i}(x,y) &:= -x_i + e_i^T x_u \geq 0, \quad \forall i \in \{1,\ldots,n_c\}, \\ g_{\tilde{m}+n_c+i}(x,y) &:= x_i - e_i^T x_l \geq 0, \quad \forall i \in \{1,\ldots,n_c\}, \\ g_{\tilde{m}+2n_c+i}(x,y) &:= -y_i + e_i^T y_u \geq 0, \quad \forall i \in \{1,\ldots,n_i\}, \\ g_{\tilde{m}+2n_c+n_i+i}(x,y) &:= y_i - e_i^T y_l \geq 0, \quad \forall i \in \{1,\ldots,n_i\}. \end{aligned} \tag{3.2}$$

Nevertheless, we still denote the feasible domains induced by the bounds on the continuous and integer variables by X and Y, i.e.,

$$X := \{x \in \mathbb{R}^{n_c} : g_{\tilde{m}+i}(x,y) \geq 0,\ i \in \{1,\ldots,2n_c\}\}, \tag{3.3}$$

$$Y := \{y \in \mathbb{N}^{n_i} : g_{\tilde{m}+2n_c+i}(x,y) \geq 0,\ i \in \{1,\ldots,2n_i\}\}. \tag{3.4}$$

To be consistent with the notation of the previous chapters, we define the number of constraints to be $m := \tilde{m} + 2n_c + 2n_i$ and extend the index set $\mathbb{J}$ accordingly. Note, that the relaxation of the set Y is denoted by $Y_{\mathbb{R}}$.

The requirements for proving convergence are subsumed in Assumption 3.1 later on in this chapter. The main restriction is, that the objective function $f(x,y)$ is required to be convex and the constraints $g_j(x,y)$ need to be concave for all $j \in \mathbb{J}$ on the relaxation of the feasible domain described by $X \times Y_{\mathbb{R}}$.

Furthermore, we denote by M_x an upper bound on the distance between two values $\hat{x},\ \tilde{x} \in X$, i.e., $\|\hat{x} - \tilde{x}\|_2 \leq M_x$ holds for all $\hat{x},\ \tilde{x} \in X$. From a practical point of view, M_x corresponds to the maximal distance between two bounds, i.e.,

$$M_x := \sqrt{n_c} \max_{i \in \{1,\ldots,n_c\}} \{(x_u)_i - (x_l)_i\}. \tag{3.5}$$

As presented in Section 2.5, linear outer approximation algorithms guarantee global optimality by the successive solution of MILP master problems. The master problem is a linear relaxation of the original convex MINLP, which is refined in each iteration yielding a monotone increasing sequence of lower bounds on the optimal objective value of MINLP (3.1). The new algorithm to be proposed in this section guarantees convergence properties for convex MINLPs by the master problem in a similar way.

As a consequence, we have to gain the desired increase in efficiency and robustness by modifying the remaining part of the linear outer approximation algorithm. The basic idea is to look for improving integer values instead of fixing the integer variables. This implies that the integer variables are allowed to vary not only during the solution of the master problem. Therefore, within this chapter both the continuous and the integer variables depend on the iteration index k.

As motivated in Section 2.5, it is profitable to apply the trust region method of Yuan, i.e., Algorithm 2.1, within a linear outer approximation method such as Algorithm 2.2

for solving NLP(y^k) for a given iterate $y^k \in Y$. In the reminder of this chapter NLP(y^k) is given by

$$\begin{aligned} & x \in \mathbb{R}^{n_c} : \\ & \quad \min \quad f(x, y^k) \\ & \quad \text{s.t.} \quad g_j(x, y^k) \ \geq \ 0, \quad \forall\, j \in \mathbb{J}. \end{aligned} \tag{3.6}$$

As established by Corollary 2.3 and 2.5 in Section 2.5, applying Yuan's trust region method has the advantage, that we need not distinguish between solving NLP(y^k) given by (3.6) and the feasibility problem F(y^k) for some fixed $y^k \in Y$, if F(y^k) is given by

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ \eta \in \mathbb{R}_+ : \\ & \min \quad \eta \\ & \text{s.t.} \quad g_j(x, y^k) + \eta \ \geq \ 0, \quad \forall j \in \mathbb{J}. \end{aligned} \tag{3.7}$$

Note, that otherwise, F(y^k) needs to be solved in addition, whenever an infeasible integer value $y \notin V$ is encountered, where the set V is given by

$$V \ = \ \{y \in Y : \exists\, x \in X \text{ with } g_j(x, y) \geq 0,\ \forall\, j \in \mathbb{J}\}. \tag{3.8}$$

See Fletcher and Leyffer [50] for a more general formulation of F(y^k).

In order to identify solutions of F(y^k), the subsequent linear program, denoted by LPF(x^k, y^k), is considered, see Corollary 2.4 relating the KKT-conditions of F(y^k) and LPF(x^k, y^k). It was already introduced in (2.90).

$$\begin{aligned} & (d_F)_x \in \mathbb{R}^{n_c},\ \eta \in \mathbb{R}_+ : \\ & \min \quad \eta \\ & \text{s.t.} \quad g_j(x^k, y^k) + \nabla_x g_j(x^k, y^k)^T (d_F)_x + \eta \ \geq \ 0, \quad \forall j \in \mathbb{J}. \end{aligned} \tag{3.9}$$

The solution of LPF(x^k, y^k) given by (3.9) is denoted by $((d_F^k)_x, \eta^k)$. In addition, we introduce $d_F := \begin{pmatrix} (d_F)_x \\ 0 \end{pmatrix}$, and extend the solution $((d_F^k)_x, \eta^k)$ to (d_F^k, η^k), with $d_F^k \in \mathbb{R}^n$ given by $d_F^k := \begin{pmatrix} (d_F^k)_x \\ 0 \end{pmatrix}$.

Solutions of NLP(y^k) also need to be identified. This task is directly established by Yuan's trust-region algorithm, see the subsequent Corollary 3.1:

Applying trust region Algorithm 2.1 for solving NLP(y^k) given by (3.6) yields the following continuous subproblem at the current iterate (x^k, y^k). It approximates the L_∞-penalty function of the continuous nonlinear program NLP(y^k), see Section 2.2.

$$\begin{aligned} & (d_c)_x \in \mathbb{R}^{n_c} : \\ & \quad \min \quad \Phi_c^k((d_c)_x) \\ & \quad \text{s.t.} \quad \|(d_c)_x\|_\infty \ \leq \ \Delta_c^k, \end{aligned} \tag{3.10}$$

where the objective function is given by

$$\begin{aligned} \Phi_c^k((d_c)_x) \;&:=\; \nabla_x f(x^k,y^k)^T(d_c)_x + \frac{1}{2}(d_c)_x^T B_c^k (d_c)_x \\ &+\; \sigma^k \|(g(x^k,y^k) + [\nabla_x g(x^k,y^k)]^T (d_c)_x)^-\|_\infty . \end{aligned} \tag{3.11}$$

$B_c^k \in \mathbb{R}^{n_c \times n_c}$ is the upper left sub-matrix of the $n \times n$-matrix B^k, which is symmetric and positive definite. B^k is possibly a Quasi-Newton approximation of the Hessian matrix of the Lagrangian function, see Definition 2.4 and 2.7. $\sigma^k \in \mathbb{R}_+$ is the penalty parameter of the L_∞ penalty function. $(.)^-$ is defined analogue to Definition 2.9 and $\Delta_c^k \in \mathbb{R}_+$ is the trust region radius, i.e., it is equivalent to Δ^k in Algorithm 2.1. Furthermore, we define $d_c := \begin{pmatrix} (d_c)_x \\ 0 \end{pmatrix} \in \mathbb{R}^{n_c+n_i}$, where 0 is the vector of all zeros of dimension n_i. According to Section 2.2 the solution of (3.10) is denoted by $(d_c^k)_x$ and therefore d_c^k is defined by $d_c^k := \begin{pmatrix} (d_c^k)_x \\ 0 \end{pmatrix} \in \mathbb{R}^{n_c+n_i}$, where 0 is the vector of all zeros of dimension n_i.

Problem (3.10) is equivalent to the quadratic program

$$\begin{aligned} &(d_c)_x \in \mathbb{R}^{n_c},\ \eta_c \in \mathbb{R}_+ : \\ &\begin{aligned} \min \quad & \nabla_x f(x^k,y^k)^T(d_c)_x + \tfrac{1}{2}(d_c)_x^T B_c^k (d_c)_x + \sigma^k \eta_c \\ \text{s.t.} \quad & \eta_c + g_j(x^k,y^k) + \nabla_x g_j(x^k,y^k)^T (d_c)_x \;\geq\; 0, \quad j = 1,\dots,m, \\ & \|(d_c)_x\|_\infty \;\leq\; \Delta_c^k, \end{aligned} \end{aligned} \tag{3.12}$$

see Yuan [112]. QP (3.12) is denoted by $QP(x^k,y^k)$, since it depends on the modeling point (x^k,y^k) determining function and gradient values. Furthermore, (3.12) is a strictly convex quadratic program and therefore can be solved efficiently.

The subsequent corollary relates the KKT-conditions of a KKT-point $(\eta_c^k, (d_c^k)_x, (\lambda_c^k, \lambda_{\eta_c}^k))$ of $QP(x^k,y^k)$ given by (3.12) to those of $NLP(y^k)$ given by (3.6).

Corollary 3.1. *For some $y^k \in Y$ and a sufficiently large value of the penalty parameter σ^k, $(\bar{x}_{y^k}, \bar{\lambda})$ is a KKT-point of NLP(y^k) given by* (3.6)*, if and only if $(\eta_c^k, (d_c^k)_x, (\lambda_c^k, \lambda_{\eta_c}^k))$ is a KKT-point of QP($\bar{x}_{y^k}, y^k$) given by* (3.12) *with $(d_c^k)_x = 0$, $\eta_c^k = 0$.*

Proof. The KKT-conditions of $QP(\bar{x}_{y^k}, y^k)$ given by (3.12) for the KKT-point $(\eta_c^k, (d_c^k)_x, (\lambda_c^k, \lambda_{\eta_c}^k))$ are determined by the subsequent formulas, where the trust region constraint

$$\|(d_c)_x\|_\infty \;\leq\; \Delta_c^k \tag{3.13}$$

is neglected, since it is not active for $(d_c^k)_x = 0$ and $\Delta_c^k > 0$. Note, that λ_c^k contains the Lagrangian multipliers $(\lambda_c^k)_j,\ j \in \mathbb{J}$ associated with the constraints

$$\eta_c + g_j(x^k,y^k) + \nabla_x g_j(x^k,y^k)^T (d_c)_x \;\geq\; 0,\ j \in \mathbb{J}. \tag{3.14}$$

The Lagrangian multiplier $\lambda_{\eta_c}^k$ is associated with the non-negativity condition for η_c. We get the following KKT-conditions:

$$\begin{aligned}
\nabla_x f(\bar{x}_{y^k}, y^k) + B_c^k (d_c^k)_x - \sum_{j=1}^{m} (\lambda_c^k)_j \nabla_x g_j(\bar{x}_{y^k}, y^k) &= 0, \\
\sigma^k &= \sum_{j=1}^{m} (\lambda_c^k)_j + \lambda_{\eta_c}^k \\
\eta_c^k + g_j(\bar{x}_{y^k}, y^k) + \nabla_x g_j(\bar{x}_{y^k}, y^k)^T (d_c^k)_x &\geq 0, \ \forall j \in \mathbb{J}, \\
\eta_c^k &\geq 0, \\
(\lambda_c^k)_j &\geq 0, \ \forall j \in \mathbb{J}, \\
\lambda_{\eta_c}^k &\geq 0, \\
(\lambda_c^k)_j (\eta_c^k + g_j(\bar{x}_{y^k}, y^k) + \nabla_x g_j(\bar{x}_{y^k}, y^k)^T (d_c^k)_x) &= 0, \ \forall j \in \mathbb{J}, \\
\eta_c^k \lambda_{\eta_c} &= 0.
\end{aligned} \tag{3.15}$$

Case 1 '⇒': $(\bar{x}_{y^k}, \bar{\lambda})$ is a KKT-point of NLP(y^k) given by (3.6):
$\Rightarrow (\eta_c^k, (d_c^k)_x, (\lambda_c^k, \lambda_{\eta_c}^k))$ with $\eta_c^k := 0$, $(d_c^k)_x := 0$ and $(\lambda_c^k)_j := \bar{\lambda}_j$, $j \in \mathbb{J}$ and $\lambda_{\eta_c}^k := \sigma^k - \sum_{j \in \mathbb{J}} \bar{\lambda}_j$ satisfies (3.15), since $(\bar{x}_{y^k}, \bar{\lambda})$ satisfies the subsequent KKT-condition of NLP(y^k):

$$\begin{aligned}
\nabla_x f(\bar{x}_{y^k}, y^k) - \sum_{j \in \mathbb{J}} \bar{\lambda}_j \nabla_x g_j(\bar{x}_{y^k}, y^k) &= 0, \\
g_j(\bar{x}_{y^k}, y^k) &\geq 0, \quad j \in \mathbb{J}, \\
\bar{\lambda}_j g_j(\bar{x}_{y^k}, y^k) &= 0, \quad j \in \mathbb{J}, \\
\bar{\lambda}_j &\geq 0, \quad j \in \mathbb{J}.
\end{aligned} \tag{3.16}$$

Case 2 '⇐': $(\eta_c^k, (d_c^k)_x, (\lambda_c^k, \lambda_{\eta_c}^k))$ is a KKT-point of QP($\bar{x}_{y^k}, y^k$) with $\eta_c^k := 0$, $(d_c^k)_x := 0$ satisfying (3.15):
$\Rightarrow (\bar{x}_{y^k}, \bar{\lambda})$ with $\bar{\lambda}_j := (\lambda_c^k)_j$, $j \in \mathbb{J}$ satisfies the KKT-conditions of NLP(y^k) stated in (3.16) due to (3.15).

□

A property of the linear outer approximation Algorithm 2.2 is, that it terminates after a finite number of iterations due to the finiteness of the set Y. In principle, this can only be assured, if the solution process of NLP(y^k) given by (3.6) or F(y^k) given by

(3.7) respectively, also terminates after a finite number of iterations in each iteration of a linear outer approximation method. This topic is usually neglected in existing literature. To be able to prove finite termination of the algorithm to be proposed in this section, we introduce a ε-stationary-point of NLP(y^k) and F(y^k) for fixed $y^k \in Y$ in the subsequent definition.

Definition 3.1. *A point (x^k, λ^k) with $y^k \in V$ is a ε-stationary point of NLP(y^k) subject to a tolerance $\varepsilon > 0$, if the following approximations of the KKT-conditions of NLP(y^k) are satisfied:*

$$\begin{aligned} \|\nabla_x f(x^k, y^k) - \sum_{j \in \mathbb{J}} \lambda_j^k \nabla_x g_j(x^k, y^k)\|_2 &\leq \varepsilon, \\ g_j(x^k, y^k) &\geq -\varepsilon, \quad \forall j \in \mathbb{J} \\ |\lambda_j^k \left(g_j(x^k, y^k)\right)| &\leq \varepsilon, \quad \forall j \in \mathbb{J}, \\ \lambda_j^k &\geq -\varepsilon, \quad \forall j \in \mathbb{J}. \end{aligned} \tag{3.17}$$

Moreover, a point $(x^k, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ with $y^k \in Y \backslash V$ is a ε-stationary point of F(y^k) subject to a tolerance $\varepsilon > 0$, if the following approximations of the KKT-conditions of F(y^k) are satisfied:

$$\begin{aligned} \left\| \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \sum_{j \in \mathbb{J}} (\lambda_F^k)_j \begin{pmatrix} \nabla_x g_j(x^k, y^k) \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ \lambda_\eta^k \end{pmatrix} \right\|_2 &\leq \varepsilon, \\ |\left(g_j(x^k, y^k) + \eta^k\right) (\lambda_F^k)_j| &\leq \varepsilon, \quad \forall j \in \mathbb{J}, \\ |\eta^k \lambda_\eta^k| &\leq \varepsilon, \\ g_j(x^k, y^k) + \eta^k &\geq -\varepsilon, \quad \forall j \in \mathbb{J}, \\ \eta^k &\geq -\varepsilon, \\ (\lambda_F^k)_j &\geq -\varepsilon, \quad \forall j \in \mathbb{J}, \\ \lambda_\eta^k &\geq -\varepsilon. \end{aligned} \tag{3.18}$$

The subsequent corollary establishes the relationship of a ε-KKT point of NLP(y^k) introduced in Definition 3.1 and the solution of subproblem QP(x^k, y^k).

Corollary 3.2. *Let $((d_c^k)_x, \eta_c^k, (\lambda_c^k, \lambda_{\eta_c}^k))$ be a KKT-point of QP(x^k, y^k) with*

$$\|(d_c^k)_x\|_2 \leq \tilde{\varepsilon}, \tag{3.19}$$

$$\eta_c^k \leq \tilde{\varepsilon} \tag{3.20}$$

and $\tilde{\varepsilon} > 0$. Furthermore let

$$\|(d_c^k)_x\|_\infty < \Delta_c^k \tag{3.21}$$

hold, i.e., the trust region constraint of QP(x^k, y^k) can be neglected. Then (x^k, λ_c^k) *is a* ε*-stationary point of NLP(y^k) according to Definition 3.1 subject to an accuracy* ε *satisfying*

$$\varepsilon \geq \max\{M_B\tilde{\varepsilon}, (1+M_{\nabla g})\tilde{\varepsilon}, (1+M_{\nabla g})M_\lambda\tilde{\varepsilon}\} \tag{3.22}$$

with $\|B_c^k\|_2 \leq M_B$, $\|\nabla_x g_j(x^k, y^k)\|_2 \leq M_{\nabla g}$, $\forall j \in \mathbb{J}$ *and* $|(\lambda_c^k)_j| \leq M_\lambda$, $\forall j \in \mathbb{J}$.

Proof. **Optimality:** Since $((d_c^k)_x, \eta_c^k, (\lambda_c^k, \lambda_{\eta_c}^k))$ is a KKT-point of QP(x^k, y^k)

$$\nabla_x f(x^k, y^k) + B_c^k (d_c^k)_x - \sum_{j\in\mathbb{J}} (\lambda_c^k)_j \nabla_x g_j(x^k, y^k) = 0. \tag{3.23}$$

holds. As a consequence we obtain

$$\begin{aligned} \|\nabla_x f(x^k, y^k) - \sum_{j\in\mathbb{J}} (\lambda_c^k)_j \nabla_x g_j(x^k, y^k)\|_2 &= \|-B_c^k (d_c^k)_x\|_2 \\ &\leq \|B_c^k\|_2 \|(d_c^k)_x\|_2 \\ &\leq M_B\tilde{\varepsilon}. \end{aligned} \tag{3.24}$$

Primal Feasibility: Since $((d_c^k)_x, \eta_c^k, (\lambda_c^k, \lambda_{\eta_c}^k))$ is a KKT-point of QP(x^k, y^k)

$$\eta_c^k + g_j(x^k, y^k) + \nabla_x g_j(x^k, y^k)^T (d_c^k)_x \geq 0, \ \forall j \in \mathbb{J} \tag{3.25}$$

holds. As a consequence we obtain $\forall j \in \mathbb{J}$

$$\begin{aligned} g_j(x^k, y^k) &\geq -\eta_c^k - \nabla_x g_j(x^k, y^k)^T (d_c^k)_x \\ &\geq -\eta_c^k - |\nabla_x g_j(x^k, y^k)^T (d_c^k)_x| \\ &\geq -\eta_c^k - \|\nabla_x g_j(x^k, y^k)^T\|_2 \|(d_c^k)_x\|_2 \\ &\geq -\tilde{\varepsilon} - M_{\nabla g}\tilde{\varepsilon} \\ &\geq -(1+M_{\nabla g})\tilde{\varepsilon}. \end{aligned} \tag{3.26}$$

Dual Feasibility: Since $((d_c^k)_x, \eta_c^k, (\lambda_c^k, \lambda_{\eta_c}^k))$ is a KKT-point of QP(x^k, y^k)

$$(\lambda_c^k)_j \geq 0, \ \forall j \in \mathbb{J} \tag{3.27}$$

holds.

Complementarity: Since $((d_c^k)_x, \eta_c^k, (\lambda_c^k, \lambda_{\eta_c}^k))$ is a KKT-point of QP(x^k, y^k)

$$(\lambda_c^k)_j \left(\eta_c^k + g_j(x^k, y^k) + \nabla_x g_j(x^k, y^k)^T (d_c^k)_x\right) = 0, \ \forall j \in \mathbb{J} \tag{3.28}$$

holds. We obtain $\forall j \in \mathbb{J}$

$$\begin{aligned}
|(\lambda_c^k)_j g_j(x^k, y^k)| &= |(\lambda_c^k)_j \left(-\eta_c^k - \nabla_x g_j(x^k, y^k)^T (d_c^k)_x\right)| \\
&\leq |(\lambda_c^k)_j| |-\eta_c^k - \nabla_x g_j(x^k, y^k)^T (d_c^k)_x| \\
&\leq |(\lambda_c^k)_j| |\eta_c^k + \nabla_x g_j(x^k, y^k)^T (d_c^k)_x| \\
&\leq |(\lambda_c^k)_j| \left(|\eta_c^k| + |\nabla_x g_j(x^k, y^k)^T (d_c^k)_x|\right) \\
&\leq |(\lambda_c^k)_j| \left(|\eta_c^k| + \|\nabla_x g_j(x^k, y^k)^T\|_2 \|(d_c^k)_x\|_2\right) \\
&\leq M_\lambda \left(\tilde{\varepsilon} + M_{\nabla g} \tilde{\varepsilon}\right) \\
&\leq (1 + M_{\nabla g}) M_\lambda \tilde{\varepsilon}.
\end{aligned} \tag{3.29}$$

As a consequence (x^k, λ_c^k) is a ε-stationary point of NLP(y^k) according to Definition 3.1 subject to an accuracy ε satisfying

$$\varepsilon \geq \max\{M_B \tilde{\varepsilon}, (1 + M_{\nabla g})\tilde{\varepsilon}, (1 + M_{\nabla g}) M_\lambda \tilde{\varepsilon}\}. \tag{3.30}$$

□

The subsequent corollary establishes the relationship of a ε-KKT point of F(y^k) introduced in Definition 3.1 and the solution of subproblem LPF(x^k, y^k).

Corollary 3.3. *Let* $((d_F^k)_x, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ *be a KKT-point of LPF(*x^k, y^k*) with*

$$\|(d_F^k)_x\|_2 \leq \tilde{\varepsilon}, \tag{3.31}$$

$$\eta^k > \tilde{\varepsilon} \tag{3.32}$$

and $\tilde{\varepsilon} > 0$*. Then* $(x^k, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ *is a* ε*-stationary point of F(*y^k*) according to Definition 3.1 subject to an accuracy* ε *satisfying*

$$\varepsilon \geq \max\{M_{\nabla g} \tilde{\varepsilon}, M_{\nabla g} M_\lambda \tilde{\varepsilon}\} \tag{3.33}$$

with $\|\nabla_x g_j(x^k, y^k)\|_2 \leq M_{\nabla g}, \ \forall j \in \mathbb{J}$ *and* $|(\lambda_F^k)_j| \leq M_\lambda, \ \forall j \in \mathbb{J}$.

Proof. **Optimality:** Since $((d_F^k)_x, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ is a KKT-point of LPF(x^k, y^k)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} - \sum_{j \in \mathbb{J}} (\lambda_F^k)_j \begin{pmatrix} \nabla_x g_j(x^k, y^k) \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ \lambda_\eta^k \end{pmatrix} = 0 \tag{3.34}$$

holds.

Primal Feasibility: Since $((d_F^k)_x, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ is a KKT-point of LPF(x^k, y^k)

$$g_j(x^k, y^k) + \nabla_x g_j(x^k, y^k)^T (d_F^k)_x + \eta^k \geq 0, \ \forall j \in \mathbb{J}, \tag{3.35}$$

holds. As a consequence we obtain

$$\begin{aligned} g_j(x^k, y^k) + \eta^k &\geq -\nabla_x g_j(x^k, y^k)^T (d_F^k)_x \\ &\geq -|\nabla_x g_j(x^k, y^k)^T (d_F^k)_x| \\ &\geq -\|\nabla_x g_j(x^k, y^k)\|_2 \|(d_F^k)_x\|_2 \\ &\geq -M_{\nabla g}\tilde{\varepsilon}, \ \forall j \in \mathbb{J}. \end{aligned} \tag{3.36}$$

In addition

$$\eta^k \geq 0 \tag{3.37}$$

is satisfied due to (3.32).

Dual Feasibility: Since $((d_F^k)_x, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ is a KKT-point of LPF(x^k, y^k)

$$\begin{aligned} (\lambda_F^k)_j &\geq 0, \ \forall j \in \mathbb{J}, \\ \lambda_\eta^k &\geq 0 \end{aligned} \tag{3.38}$$

holds.

Complementarity: Since $((d_F^k)_x, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ is a KKT-point of LPF(x^k, y^k)

$$\begin{aligned} \left(g_j(x^k, y) + \nabla_x g_j(x^k, y)^T (d_F^k)_x + \eta^k\right) (\lambda_F^k)_j &= 0, \ \forall j \in \mathbb{J}, \\ \eta^k \lambda_\eta^k &= 0 \end{aligned} \tag{3.39}$$

holds. We obtain

$$\begin{aligned} |\left(g_j(x^k, y) + \eta^k\right) (\lambda_F^k)_j| &= |-\left(\nabla_x g_j(x^k, y)^T (d_F^k)_x\right) (\lambda_F^k)_j| \\ &\leq \|\nabla_x g_j(x^k, y)\|_2 \|(d_F^k)_x\|_2 |(\lambda_F^k)_j| \\ &\leq M_{\nabla g}\tilde{\varepsilon} M_\lambda, \ \forall j \in \mathbb{J}. \end{aligned} \tag{3.40}$$

As a consequence $(x^k, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ is a ε-stationary point of $F(y^k)$ according to Definition 3.1 subject to an accuracy ε satisfying

$$\varepsilon \geq \max\{M_{\nabla g}\tilde{\varepsilon}, M_{\nabla g} M_\lambda \tilde{\varepsilon}\} \tag{3.41}$$

□

To ease the readability, the definition of the master problem of the linear outer approximation method introduced in Chapter 2 is repeated. To point out the dependencies,

it is denoted by MILP($T_\varepsilon^k, S_\varepsilon^k, \hat{\eta}^k, \varepsilon_{OA}$), where T_ε^k, S_ε^k, and $\hat{\eta}^k$ are specified below.

$$
\begin{aligned}
& x \in \mathbb{R}^{n_c},\ y \in \mathbb{N}^{n_i},\ \eta \in \mathbb{R}: \qquad (3.42)\\
& \min \quad \eta \\
& \text{s.t.} \quad f(x^i, y^i) + \nabla_{x,y} f(x^i, y^i)^T \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} \leq \eta, \quad \forall (x^i, y^i) \in T_\varepsilon^k, \\
& \qquad g(x^i, y^i) + [\nabla_{x,y} g(x^i, y^i)]^T \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix} \geq 0, \quad \forall (x^i, y^i) \in T_\varepsilon^k, \\
& \qquad g(x^j, y^j) + [\nabla_{x,y} g(x^j, y^j)]^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} \geq 0, \quad \forall (x^j, y^j) \in S_\varepsilon^k, \\
& \qquad \eta \leq \hat{\eta}^k - \varepsilon_{OA}.
\end{aligned}
$$

The sets T^k and S^k introduced in (2.71) and (2.72) need to be adapted based on Definition 3.1 yielding

$$
T_\varepsilon^k \subset \left\{ \begin{array}{l} (x^i, y^i) \in \mathbb{R}^{n_c} \times \{y^1, \ldots, y^k\}: \\ (x^i, \lambda^i) \text{ is } \varepsilon - \text{stationary point of NLP}(y^i) \end{array} \right\} \qquad (3.43)
$$

$$
S_\varepsilon^k \subset \left\{ \begin{array}{l} (x^j, y^j) \in \mathbb{R}^{n_c} \times \{y^1, \ldots, y^k\}: \\ (x^j, \eta^j, (\lambda_F^j, \lambda_\eta^j)) \text{ is } \varepsilon - \text{stationary point of F}(y^j) \end{array} \right\}. \qquad (3.44)
$$

Similar to well-known linear outer approximation methods, such as Algorithm 2.2, the sets T_ε^k and S_ε^k are updated, such that they contain one out of the infinitely many ε-stationary points of each NLP(y^i) given by (3.6) with $i \leq k$ or F(y^j) given by (3.7) with $j \leq k$, that was obtained in previous iterations.

$\hat{\eta}^k$ defines an upper bound on η given by

$$
\hat{\eta}^k := \min\{f(x^i, y^i): (x^i, y^i) \in T_\varepsilon^k\}, \qquad (3.45)
$$

where T_ε^k is defined by (3.43).

Note, that the bounds x_l, x_u, y_l and y_u on continuous and integer variables, which are a subset of the constraints g_j, $j \in \mathbb{J}$, are satisfied at the solution of MILP($T_\varepsilon^k, S_\varepsilon^k, \hat{\eta}^k, \varepsilon_{OA}$), since

$$
\begin{aligned}
g_{\tilde{m}+i}(x^j, y^j) + \nabla_{x,y} g_{\tilde{m}+i} \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} &= -x_i^j + e_i^T x_u - x_i + x_i^j, \\
&= e_i^T x_u - x_i, \\
&\geq 0
\end{aligned} \qquad (3.46)
$$

holds for each upper bound on any continuous variable x_i, $i \in \{1, \ldots, n_c\}$ for each $(x^j, y^j) \in S_\varepsilon^k$, or T_ε^k respectively. The same is valid for all lower bounds on continuous variables and for upper and lower bounds on integer variables as well.

Up to now we motivated the part of the new algorithm, that is derived from linear outer approximation methods, such as Algorithm 2.2. As mentioned at the beginning

of this chapter, we want to combine the linear outer approximation approach with ideas implemented in MISQP, which is reviewed in Section 2.10. Since the algorithmic concept of MISQP is based on sequence of mixed-integer quadratic approximations, we focus now on the integration of a MIQP approximation into a linear outer approximation approach.

The L_∞-penalty function corresponding to NLP(y) given by (3.6) can also be associated with the continuous relaxation of MINLP (3.1), which is given by

$$\begin{aligned} x \in \mathbb{R}^{n_c},\ y \in \mathbb{R}^{n_i} :& \\ \min \quad & f(x,y) \\ \text{s.t.} \quad & g_j(x,y) \ \geq \ 0, \quad \forall\, j \in \mathbb{J}. \end{aligned} \tag{3.47}$$

Inspired by MISQP, we apply the algorithm of Yuan, see Yuan [112], to solve the continuous relaxation of MINLP (3.1) and replace the corresponding continuous subproblems by a mixed-integer formulation. This mixed-integer problem depends on the current iteration point (x^k, y^k) and is given by

$$\begin{aligned} d_i \in \mathbb{R}^{n_c} \times \mathbb{N}^{n_i} :& \\ \min \quad & \Phi_i^k(d_i) \\ \text{s.t.} \quad & \|d_i\|_\infty \ \leq \ \Delta_i^k, \end{aligned} \tag{3.48}$$

where the objective function is given by

$$\begin{aligned} \Phi_i^k(d_i) \ := \quad & \nabla_{x,y} f(x^k,y^k)^T d_i + \tfrac{1}{2} d_i^T B^k d_i \\ + \quad & \sigma^k \|(g(x^k,y^k) + [\nabla_{x,y} g(x^k,y^k)]^T d_i)^-\|_\infty, \end{aligned} \tag{3.49}$$

with $d_i = \begin{pmatrix} (d_i)_x \\ (d_i)_y \end{pmatrix}$ and $(d_i)_x \in \mathbb{R}^{n_c}$ and $(d_i)_y \in \mathbb{N}^{n_i}$. $B^k \in \mathbb{R}^{n \times n}$ is a symmetric and positive definite matrix, possibly a Quasi-Newton approximation of the Hessian matrix of the Lagrangian function, see Definition 2.4 and 2.7. $\sigma^k \in \mathbb{R}_+$ is the penalty parameter of the L_∞-penalty function. $(.)^-$ is defined analogue to Definition 2.9 and $\Delta_i^k \in \mathbb{R}_+$ is the trust region radius.

Analogue to QP(x^k, y^k), problem (3.48) is equivalent to the mixed-integer quadratic program denoted by MIQP(x^k, y^k)

$$\begin{aligned} & d_i \in \mathbb{R}^{n_c} \times \mathbb{N}^{n_i},\ \eta_i \in \mathbb{R}_+ : \\ & \min \quad \nabla_{x,y} f(x^k,y^k)^T d_i + \tfrac{1}{2} d_i^T B^k d_i + \sigma^k \eta_i \\ & \text{s.t.} \quad \eta_i + g_j(x^k,y^k) + \nabla_{x,y} g_j(x^k,y^k)^T d_i \ \geq \ 0, \quad j = 1,\ldots,m, \\ & \qquad\qquad \|d_i\|_\infty \ \leq \ \Delta_i^k. \end{aligned} \tag{3.50}$$

The solution of (3.50) is denoted by (d_i^k, η_i^k). The search direction with respect to the integral variables $y^k \in Y$ is restricted to integral values, i.e., $(d_i)_y \in \mathbb{N}^{n_i}$. As a consequence, integrality is satisfied for $y^k + (d_i^k)_y$, i.e., $y^k + (d_i^k)_y \in \mathbb{N}^{n_i}$.

The main idea of the new algorithm is to compare the search step determined by MIQP(x^k, y^k) with that determined by QP(x^k, y^k) with respect to the value of the L_∞-penalty function and to choose the better one, i.e., the one with a lower merit function value. As a consequence, we define an improving mixed-integer search direction as follows.

Definition 3.2. *The solution $d_i^k \in \mathbb{R}^{n_c} \times \mathbb{N}^{n_i}$ of MIQP(x^k, y^k) given by* (3.50) *is an improving mixed-integer search direction, if it satisfies the following conditions:*

1.

$$\frac{P_{\sigma^k}(x^k, y^k) - P_{\sigma^k}(x^k + (d_i^k)_x, y^k + (d_i^k)_y)}{\Phi_c^k(0) - \Phi_c^k((d_c^k)_x)} \geq 0.1, \tag{3.51}$$

2.

$$P_{\sigma^k}(x^k + (d_i^k)_x, y^k + (d_i^k)_y) < P_{\sigma^k}(x^k + (d_c^k)_x, y^k). \tag{3.52}$$

$(d_c^k)_x \in \mathbb{R}^{n_c}$ *is part of the solution* $(\eta_c^k, (d_c^k)_x)$ *of QP(x^k, y^k) given by* (3.12). $P_{\sigma^k}(x, y)$ *denotes the* L_∞*-penalty function with respect to the penalty parameter* σ^k*, see Definition 2.9. Furthermore,* Φ_c^k *is defined by* (3.11).

Definition 3.2 motivates an extension of Step 2 of the linear outer approximation method described by Algorithm 2.2, that is based on two models. The first model corresponds to the continuous quadratic problem (3.12) and is called the continuous model. It is equivalent to the subproblem, that arises during the solution of NLP(y^k) by the trust region method given by Algorithm 2.1 in some iteration of a linear outer approximation method such as Algorithm 2.2. The second model represented by MIQP (3.50) is called the mixed-integer model. Analogue to MISQP, see Section 2.10, it is a mixed-integer quadratic approximation derived from MINLP (3.1).

We introduce and motivate the parameters of the algorithm. The notation of the parameters is chosen according to Yuan's trust region method given by Algorithm 2.1. In the sequel of this section, the parameters that correspond to the continuous model associated with problem (3.12) are indexed by c, while parameters of the mixed-integer model corresponding to problem (3.50) are indexed by i. Therefore, Δ_c^k denotes the trust region radius for the continuous model, while Δ_i^k is the trust region radius for the mixed-integer model. σ^k denotes the penalty parameter associated with the L_∞-penalty function. As soon as the penalty parameter is larger than an upper bound $\bar{\sigma} \in \mathbb{R}_+$, the determination of mixed-integer quadratic search steps is omitted. Note, that only one penalty parameter is necessary, since the same L_∞-penalty function is associated with the continuous model represented by problem (3.12) and the mixed-integer model associated with problem (3.50).

The reduction of the merit function predicted by the corresponding model, which is either $QP(x^k, y^k)$ given by (3.12) or $MIQP(x^k, y^k)$ given by (3.50), is given by

$$\Phi_c^k(0) - \Phi_c^k((d_c^k)_x) \quad \text{or} \quad \Phi_i^k(0) - \Phi_i^k(d_i^k),$$

respectively, where Φ_c^k is defined in (3.11) and Φ_i^k is specified in (3.49). The reduction obtained by the solution $((d_c^k)_x, \eta_c^k)$ of $QP(x^k, y^k)$ and (d_i^k, η_i^k) of $MIQP(x^k, y^k)$, see (3.12) and (3.50), with respect to the penalty function P_{σ^k}, see Definition 2.9, can be evaluated by

$$P_{\sigma^k}(x^k, y^k) - P_{\sigma^k}(x^k + (d_c^k)_x, y^k)$$

or

$$P_{\sigma^k}(x^k, y^k) - P_{\sigma^k}(x^k + (d_i^k)_x, y^k + (d_i^k)_y).$$

By comparing both quantities introducing r_c^k and r_i^k

$$r_c^k := \frac{P_{\sigma^k}(x^k, y^k) - P_{\sigma^k}(x^k + (d_c^k)_x, y^k)}{\Phi_c^k(0) - \Phi_c^k((d_c^k)_x)}, \tag{3.53}$$

$$r_i^k := \frac{P_{\sigma^k}(x^k, y^k) - P_{\sigma^k}(x^k + (d_i^k)_x, y^k + (d_i^k)_y)}{\Phi_i^k(0) - \Phi_i^k(d_i^k)}, \tag{3.54}$$

the precision of the predicted reduction of the merit function of the continuous and the integer model can be measured.

Based on these ideas, we propose an extension of a linear outer approximation method. Since the algorithm relies on the successive solution of mixed-integer quadratic programming problems (3.50), it is called MIQP-supported linear outer approximation (MIQPSOA).

Before we describe MIQPSOA in detail, we provide a brief overview to ease the readability and understanding. The brief outline is aligned with Figure 3.1, which provides a graphical representation of MIQPSOA and its components. As already described, MIQPSOA is based on the trust region method of Yuan, see Algorithm 2.1. The corresponding components are represented in green in Figure 3.1. Combining the green and blue components, MIQPSOA yields a linear outer approximation method, such as Algorithm 2.2. The red-colored components represent the new components, that are motivated by MISQP, see Section 2.10. In addition the algorithm possesses some coordination and decision steps.

Step 1: Within the **Initialization** the algorithmic parameters including the tolerance ε_{OA} and the sets $T_\varepsilon^{-1}, S_\varepsilon^{-1}$ as well as the best known solution and the corresponding objective value f^* are initialized. Furthermore, a starting point $(x^0, y^0) \in X \times Y$ together with the corresponding function and gradient values is provided.

Step 2: After the initialization is finished, the internal iteration loop starts by solving **$QP(x^k, y^k)$**, where k denotes the current iteration.

Step 3: Depending on the value of the penalty parameter the linear subproblem **LPF**(x^k, y^k) derived from $F(y^k)$, see (3.7) is solved. Alternatively the mixed-integer quadratic program **MIQP**(x^k, y^k) is solved, if the flag on^k_{MIQP} possesses the value 1.

Step 4: The subsequent coordination step executes the **Search Step Selection**. 4 different possibilities for selecting the search step arise:

If an improving mixed-integer search direction according to Definition 3.2 was obtained by MIQP(x^k, y^k), **then** it is chosen to be the search step.

Else if the current iterate is a ε-stationary-point of NLP(y^k) or $F(y^k)$ according to Definition 3.1, **then** the search step is determined by the outer approximation master problem MILP$(T^k_\varepsilon, S^k_\varepsilon, f^*, \varepsilon_{OA})$, see below.

Else if the solution d^k_c of QP(x^k, y^k) provides improvement with respect to the L_∞-penalty function, measured by r^k_c (3.53), **then** the subsequent iterate is obtained by adding d^k_c.

Else no step is performed and the trust region radius is decreased.

Step 5: If the search step is determined by the solution of either QP(x^k, y^k) or MIQP(x^k, y^k) a **Parameter Update** is performed. This affects among others the continuous trust region radius and the penalty parameter.

Step 6: If the search step is to be determined by the solution of the outer approximation master problem, **MILP**$(T^k_\varepsilon, S^k_\varepsilon, f^*, \varepsilon_{OA})$ is solved after an update of the sets $T^k_\varepsilon, S^k_\varepsilon$. If MILP$(T^k_\varepsilon, S^k_\varepsilon, f^*, \varepsilon_{OA})$ is infeasible, then MINLP (3.1) is solved.

Step 7: If a search step was performed **Gradients** are evaluated and the next iteration loop is started.

Note, that whenever the problem-functions f and g are evaluated at some point (x^k, y^k), the subsequent test checks, if the current incumbent can be updated:

Update current best solution

$$(x^*, y^*) := (x^k, y^k) \tag{3.55}$$

$$f^* := f(x^k, y^k), \tag{3.56}$$

if

$$\|g(x^k, y^k)^-\|_\infty \leq \varepsilon \tag{3.57}$$

$$f(x^k, y^k) < f^* \tag{3.58}$$

holds. In the initial step, the best known solution (x^*, y^*) is initialized as follows:

$(x^*, y^*) := (x^0, y^0)$ and

$$f^* := \begin{cases} \infty, & \text{if } \|g(x^0, y^0)^-\|_\infty > \varepsilon \\ f(x^0, y^0), & \text{if } \|g(x^0, y^0)^-\|_\infty \leq \varepsilon \end{cases}. \tag{3.59}$$

Algorithm 3.1. *MIQPSOA*

1. **Initialization***:*
   Let $x^0 \in X$, $y^0 \in Y$ *be starting values and* **define** *the parameters* $\Delta_c^0 > 0$, $\Delta_i^0 \geq 1$, $B^0 \in \mathbb{R}^{n\times n}$ *symmetric and positive definite,* $T_\varepsilon^{-1} = S_\varepsilon^{-1} := \emptyset$, $\delta^0 > 0$, $\sigma^0 > 0$, $\varepsilon_{OA} > 0$, $\varepsilon > 0$, $\bar{\sigma} \geq 0$, $on_{MIQP}^0 := 1$, $n_{oa} := 0$, $k := 0$.

   Evaluate *the functions* $f(x^0, y^0)$ *and* $g(x^0, y^0)$ *and* **determine** *gradients* $\nabla_{x,y} f(x^0, y^0)$ *and* $\nabla_{x,y} g(x^0, y^0)$ *and* **initialize** *best known solution* (x^*, y^*).

2. **QP**(x^k, y^k)*:*
   Determine *a KKT-point* $((d_c^k)_x, \eta_c^k, (\lambda_c^k, \lambda_{\eta_c}^k))$ *of QP*(x^k, y^k) *given by* (3.12).

   Evaluate $f(x^k + (d_c^k)_x, y^k)$, $g(x^k + (d_c^k)_x, y^k)$ *and* $P_{\sigma^k}(x^k + (d_c^k)_x, y^k)$, *where* P_{σ^k} *is specified in Definition 2.9.*

   Evaluate r_c^k *by* (3.53).

3. **MIQP**(x^k, y^k) *or* **LPF**(x^k, y^k)*:*

   If $on_{MIQP}^k = 1$,
   then solve *the mixed-integer program MIQP*(x^k, y^k) *given by* (3.50) *determining* (d_i^k, η_i^k).
   Evaluate $f(x^k+(d_i^k)_x, y^k+(d_i^k)_y)$, $g(x^k+(d_i^k)_x, y^k+(d_i^k)_y)$ *and* $P_{\sigma^k}(x^k+(d_i^k)_x, y^k+(d_i^k)_y)$.
   Calculate r_i^k *given by* (3.54).
   Adapt *integer trust region radius:*

$$\Delta_i^{k+1} := \begin{cases} \max\{\Delta_i^k, 4\|d_i^k\|_\infty\}, & \text{if } r_i^k > 0.9, \\ \Delta_i^k, & \text{if } 0.9 \geq r_i^k \geq 0.1, \\ \min\{\frac{1}{4}\Delta_i^k, \frac{1}{2}\|d_i^k\|_\infty\}, & \text{if } r_i^k < 0.1. \end{cases} \tag{3.60}$$

   Else if $\sigma^k > \bar{\sigma}$,
   then solve *the linear program LPF*(x^k, y^k) *given by* (3.9) *and denote the solution by* $((d_F^k)_x, \eta^k)$ *and the corresponding Lagrangian multipliers by* $(\lambda_F^k, \lambda_\eta^k)$.

4. **Search Step Selection***:*

If $on^k_{MIQP} = 1$ **and** d^k_i *is an improving search direction according to Definition 3.2,*
then set $(x^{k+1}, y^{k+1}) := (x^k + (d^k_i)_x, y^k + (d^k_i)_y)$.
GOTO *Step 5.*

Else if (x^k, λ^k_c) *is a* ε*-stationary point of NLP(*y^k*), i.e.,*

$$\|(d^k_c)_x\|_2 \leq \varepsilon, \quad \eta^k_c \leq \varepsilon, \quad \|(d^k_c)_x\|_\infty < \Delta^k_c \tag{3.61}$$

holds **and if** *in addition*

$$\begin{aligned} \varepsilon_{OA} > {} & \|\nabla_x f(x^k, y^k)\|_2 \|(d^k_c)_x\|_2 + M_x \|(d^k_c)^T_x B^k_c\|_2 \\ & + \|(d^k_c)^T_x B^k_c\|_2 \|(d^k_c)_x\|_2, \end{aligned} \tag{3.62}$$

is satisfied with M_x *defined by* (3.5),
or if $\sigma^k > \bar{\sigma}$ **and** $(x^k, \eta^k, (\lambda^k_F, \lambda^k_\eta))$ *is a* ε*-stationary point of F(*y^k*), i.e.,*

$$\|(d_F)_x\|_2 \leq \varepsilon, \quad \eta^k > \varepsilon \tag{3.63}$$

holds **and if** *in addition*

$$\|\nabla_x g_j(x^k, y^k)\|_2 \|(d^k_F)_x\|_2 < \varepsilon, \quad \forall j \in \mathbb{J}, \tag{3.64}$$

holds,
then GOTO *Step 6.*

Else if $r^k_c > 0$ *defined in* (3.53),
then set $(x^{k+1}, y^{k+1}) := (x^k + (d^k_c)_x, y^k)$ *and* $\Delta^{k+1}_i := \Delta^k_i$.

Else set

$$\begin{aligned} (x^{k+1}, y^{k+1}) &:= (x^k, y^k), & B^{k+1} &:= B^k, & \sigma^{k+1} &:= \sigma^k, \\ \delta^{k+1} &:= \delta^k, & T^k_\varepsilon &:= T^{k-1}_\varepsilon, & S^k_\varepsilon &:= S^{k-1}_\varepsilon, \\ \Delta^{k+1}_c &:= \tfrac{1}{4}\|(d^k_c)_x\|_\infty, & on^{k+1}_{MIQP} &:= on^k_{MIQP}, & \Delta^{k+1}_i &:= \Delta^k_i, \\ k &:= k+1. \end{aligned}$$

GOTO *Step 2.*

5. **Parameter Update***:*

$$\Delta^{k+1}_c := \begin{cases} \max\{\Delta^k_c, 4\|(d^k_c)_x\|_\infty\}, & \text{if } r^k_c > 0.9, \\ \Delta^k_c, & \text{if } 0.9 \geq r^k_c \geq 0.1, \\ \min\{\frac{1}{4}\Delta^k_c, \frac{1}{2}\|(d^k_c)_x\|_\infty\}, & \text{if } r^k_c < 0.1. \end{cases} \tag{3.65}$$

Choose B^{k+1}*, such that* B^{k+1} *is any symmetric, positive definite matrix.*
Set $T^k_\varepsilon := T^{k-1}_\varepsilon$, $S^k_\varepsilon := S^{k-1}_\varepsilon$.

Penalty Update with respect to Φ_c^k *defined in* (3.11)*:*

If

$$\Phi_c^k(0) - \Phi_c^k((d_c^k)_x) \;\leq\; \sigma^k \delta^k \min\left\{\Delta_c^k, \|g(x^k, y^k)^-\|_\infty\right\}, \quad (3.66)$$

then set $\sigma^{k+1} := 2\sigma^k$ *and* $\delta^{k+1} := \frac{1}{4}\delta^k$.

Else **set** $\sigma^{k+1} := \sigma^k$ *and* $\delta^{k+1} := \delta^k$.

If $\sigma^k > \bar{\sigma}$,
then set $on_{MIQP}^k := 0$.

Else **set** $on_{MIQP}^{k+1} := on_{MIQP}^k$

GOTO *Step 7.*

6. **MILP**$(T_\varepsilon^k, S_\varepsilon^k, f^*, \varepsilon_{OA})$*, i.e., outer approximation master problem:*

If (x^k, λ_c^k) *is a* ε*-stationary-point of NLP(*y^k*) given by* (3.6)*,*
then update *set* T_ε^{k-1} *given by* (3.43)*:*

$$T_\varepsilon^k \;:=\; T_\varepsilon^{k-1} \cup \{(x^k, y^k)\}, \quad S_\varepsilon^k \;:=\; S_\varepsilon^{k-1}.$$

Else *i.e.,* $(x^k, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ *is a* ε*-stationary-point of F(*y^k*) given by* (3.7)*,*
then update *set* S_ε^{k-1} *given by* (3.44)*:*

$$S_\varepsilon^k \;:=\; S_\varepsilon^{k-1} \cup \{(x^k, y^k)\}, \quad T_\varepsilon^k \;:=\; T_\varepsilon^{k-1}.$$

Solve *the linear outer approximation master problem MILP(*$T_\varepsilon^k, S_\varepsilon^k, f^*, \varepsilon_{OA}$*).*

If *linear outer approximation master problem* (3.42) *is infeasible,*
then STOP.

Else *denote the solution by* (x^{k+1}, y^{k+1}).
Evaluate $f(x^{k+1}, y^{k+1})$ *and* $g(x^{k+1}, y^{k+1})$.
Set $n_{oa} := n_{oa} + 1$, $y_{oa}^{n_{oa}} := y^{k+1}$ *and*

$$\begin{aligned} \Delta_i^{k+1} &:= \max\left\{\Delta_i^0, \Delta_i^k\right\}, \quad \Delta_c^{k+1} &:= \max\left\{\Delta_c^0, \Delta_c^k\right\}, \\ \delta^{k+1} &:= \delta^0, & \sigma^{k+1} &:= \sigma^0, \\ B^{k+1} &:= B^k. \end{aligned} \quad (3.67)$$

If $\exists\, i \in \{1, \ldots, n_{oa} - 1\}$ *with* $y^{k+1} = y_{oa}^i$,
then set $on_{MIQP}^k := 0$*, i.e., solve NLP(*y^{k+1}*) or F(*y^{k+1}*) respectively.*

Else **set** $on_{MIQP}^k := 1$.

7. **Gradients***:*
Evaluate $\nabla_{x,y} f(x^{k+1}, y^{k+1})$ *and* $[\nabla_{x,y} g(x^{k+1}, y^{k+1})]$, **set** $k := k+1$ *and* **GOTO** *Step 2.*

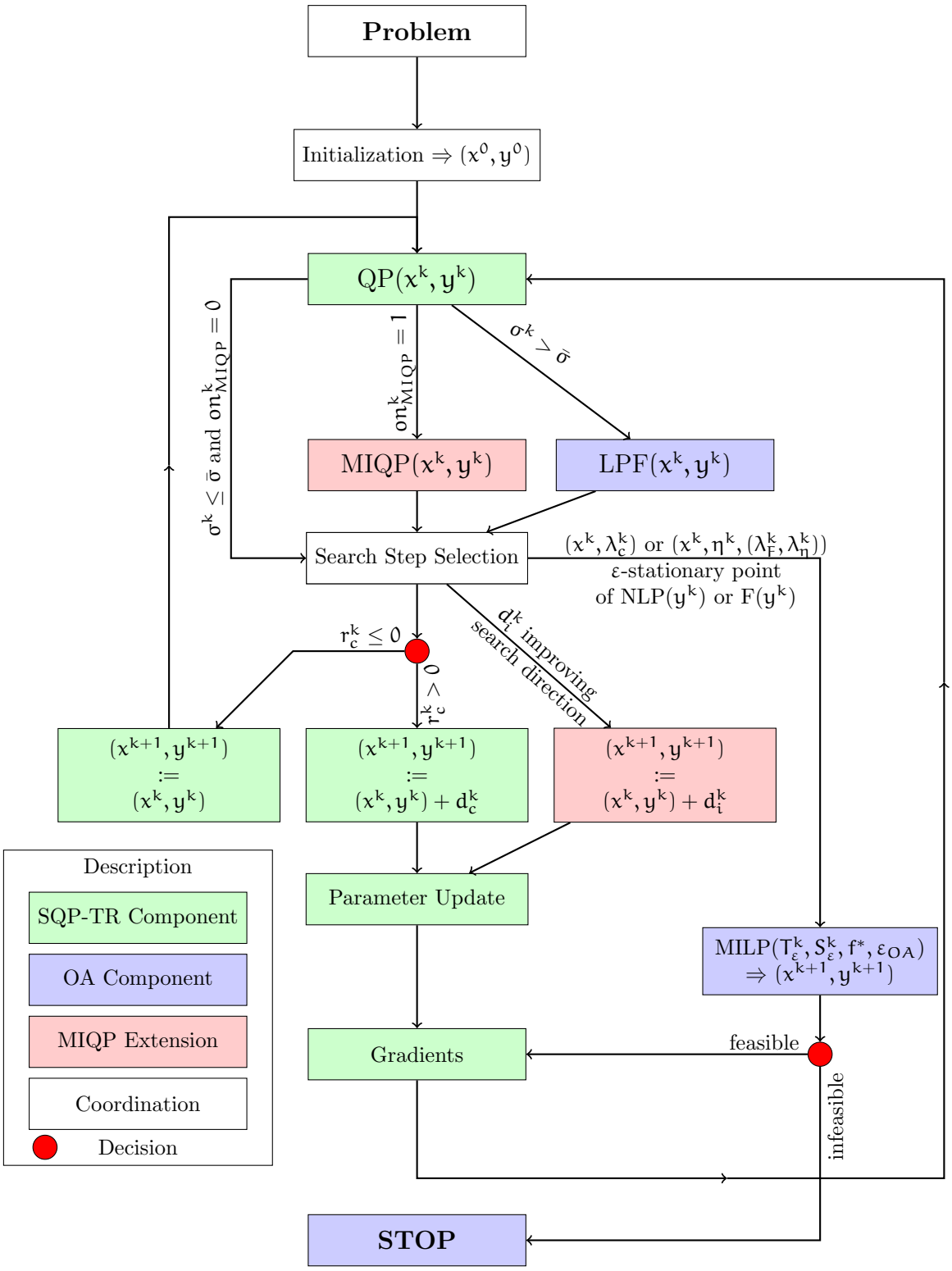

Fig. 3.1: MIQP-supported Outer Approximation

Figure 3.1 illustrates the solution process of Algorithm 3.1. The previously defined al-

gorithm is based on four different subproblems, which are $QP(x^k, y^k)$, $MIQP(x^k, y^k)$, $LPF(x^k, y^k)$ and $MILP(T_\varepsilon^k, S_\varepsilon^k, f^*, \varepsilon_{OA})$. In every iteration the continuous quadratic program $QP(x^k, y^k)$ is solved yielding a continuous search direction. This search direction is equivalent to the search direction obtained at iterate (x^k, y^k) by the trust region algorithm of Yuan stated by Algorithm 2.1, while solving $NLP(y^k)$ given by (3.6).

The parameter on_{MIQP}^k is a flag for turning on or off the calculation of the mixed-integer search direction provided by the solution of $MIQP(x^k, y^k)$ in iteration k. As shown in the remainder of this chapter, on_{MIQP}^k needs to be set to 0, i.e., $MIQP(x^k, y^k)$ is not solved, in certain situations to ensure convergence.

As will be proved below, Lemma 2.2 applies, if the penalty parameter σ^k tends towards infinity and exceeds the threshold $\bar{\sigma}$, i.e., the iteration sequence converges towards an infeasible stationary point specified in Definition 2.10. To identify these infeasible stationary points $LPF(x^k, y^k)$ is solved as soon as σ^k exceeds the threshold $\bar{\sigma}$. As a consequence, a suitable value for the parameter $\bar{\sigma}$ is $\bar{\sigma} := 10^{10}$.

The parameter ε_{OA} is the optimality tolerance needed by the outer approximation master problem $MILP(T_\varepsilon^k, S_\varepsilon^k, f^*, \varepsilon_{OA})$, in order to ensure finite termination of Algorithm 3.1. The same holds for any other linear outer approximation algorithm, such as Algorithm 2.2.

If (3.61) and (3.62) hold in Step 4, then (x^k, λ_c^k) is a ε-stationary-point of $NLP(y^k)$ given by (3.6) specified in Definition 3.1 due to Corollary 3.2. In addition (x^k, y^k) satisfies

$$\varepsilon_{OA} > \|\nabla_x f(x^k, y^k)\|_2 \|(d_c^k)_x\|_2 + M_x \|(d_c^k)_x^T B_c^k\|_2 + \|(d_c^k)_x^T B_c^k\|_2 \|(d_c^k)_x\|_2, \quad (3.68)$$

where M_x is determined by the maximal range of the continuous variables and $((d_c^k)_x, \eta_c^k, (\lambda_c^k, \lambda_{\eta_c}^k))$ is a KKT-point of $QP(x^k, y^k)$.

If $\sigma^k > \bar{\sigma}$ as well as (3.63) and (3.64) hold in Step 4 instead, then $(x^k, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ is a ε-stationary-point of $F(y^k)$ given by (3.7) specified in Definition 3.1 due to Corollary 3.3. In addition (x^k, y^k) satisfies

$$\begin{aligned} \|\nabla_x g_j(x^k, y^k)\|_2 \|(d_F^k)_x\|_2 &< \varepsilon, \quad \forall j \in \mathbb{J}, \\ \eta^k &> \varepsilon, \end{aligned} \quad (3.69)$$

where $((d_F^k)_x, \eta^k)$ is the optimal solution of $LPF(x^k, y)$ given by (3.9) and $(\lambda_F^k, \lambda_\eta^k)$ are the corresponding Lagrangian multipliers.

Note, that the values of the constants used within parameter updates, e.g. the update of the trust region radii, are taken from Yuan [112].

The convergence analysis is based on Assumptions 2.1 and 2.2, which are unified and restated here.

Assumption 3.1. *1.* $f(x, y)$ *and* $g_j(x, y)$, $j = 1, \ldots, m$ *are continuously differentiable on* $X \times Y_{\mathbb{R}}$.

2. $f(x,y)$ *is convex and* $g_j(x,y),\ j=1,\dots,m$ *are concave on* $X\times Y_{\mathbb{R}}$ *and the set* X *defined by* (3.3) *is nonempty and compact.*

3. *The linear independent constraint qualification, stated in Definition 2.6, holds at each optimal solution of problem NLP(*y*) and F(*y*) for all* $y\in Y$*.*

4. *The sequences* $\{(x^k,y^k)\}$ *and* $\{B^k\}$ *generated by the proposed algorithm are bounded* $\forall\ k$*.*

Note, that the set X defined in (3.3) is compact, due to existence of upper and lower bounds on all continuous variables.

The algorithm is designed to yield the same iteration sequence as the linear outer approximation Algorithm 2.2 under certain circumstances, where the nonlinear programs, which arise as subproblems are solved by Algorithm 2.1. This is established by the subsequent corollaries in the reminder of this section. Since the mixed-integer search steps obtained in Step 3 distinguish Algorithm 3.1 from the linear outer approximation method described by Algorithm 2.2, we skip their calculation for the moment by assuming $on^k_{MIQP}=0,\ \forall k$.

Corollary 3.4 states the equivalence of the Steps 2, 5 and 7 as well as parts of Step 4, i.e., the last **Else if**- and the **Else**-statement of Algorithm 3.1 and the trust region method of Yuan described by Algorithm 2.1.

Corollary 3.4. *Let* $on^k_{MIQP}=0,\ \forall k$ *in Algorithm 3.1 and choose the same initialization parameters* $\Delta_c^0=\Delta^0,\ B^0,\ \delta^0,\ \sigma^0$*, then Algorithm 3.1 yields the same iteration sequence as the trust region method of Yuan described by Algorithm 2.1 until either Step 6 of Algorithm 3.1 is reached or stopping criterion* (2.35) *is satisfied for Algorithm 2.1 for the same values of the initial point* (x^k,y^k)*.*

Proof. Apart from the parameters specifying conditions (3.61) and (3.62) or conditions (3.63) and (3.64) and the parameters which are associated with Step 6, the initialization in Step 1 of Algorithm 3.1 is equivalent to the one performed in Step 1 of Algorithm 2.1.

Step 2 of Algorithm 3.1 is equivalent to Step 2 of Algorithm 2.1 as long as the stopping criterion (2.35) of Algorithm 2.1 is not satisfied.

Steps 3 and 4 have no influence on the iteration sequence of Algorithm 3.1 as long as Step 6 is not reached, i.e., the conditions (3.61) and (3.62) or (3.63) and (3.64) are not satisfied:

The mixed-integer problem (3.50) is not solved in Step 3 of Algorithm 3.1, since $on^k_{MIQP}=0,\ \forall k$ holds. Furthermore, the solution of $LPF(x^k,y^k)$ given by (3.9) in Step 3 has no influence on the iteration sequence.

As the mixed-integer problem (3.50) is not solved and the conditions (3.61) and (3.62) or (3.63) and (3.64) are not satisfied, Step 4 is equivalent to Step 3 of Algorithm 2.1.

Step 5 of Algorithm 3.1 is equivalent to Step 4 of Algorithm 2.1.

Step 6 is not carried out by assumption.

Finally functions and gradients are evaluated in Step 7 of Algorithm 3.1, which corresponds to Step 5 of Algorithm 2.1.

This proves the corollary □

Based on Corollary 3.4 we can formulate the subsequent corollary establishing the relation between Algorithm 3.1 and the linear outer approximation method described by Algorithm 2.2. To point out the relationship we assume, that $d_c^k = 0$ and $\eta_c^k = 0$ or $d_F^k = 0$ and $\eta^k > 0$ holds, whenever conditions (3.61) and (3.62) or conditions (3.63) and (3.64) in Step 4 are satisfied. This means, that the optimal solution of the subproblems NLP(y^k) or F(y^k) for some $y^k \in Y$ is obtained after a finite number of iterations, which is a serious restriction in practice, that will not be required later on.

Corollary 3.5. *Let* $on_{MIQP}^k = 0,\ \forall k$*, let Assumption 3.1 hold. Furthermore assume, that either* $d_c^k = 0$ *and* $\eta_c^k = 0$ *or* $d_F^k = 0$ *and* $\eta^k > 0$ *holds, whenever conditions* (3.61) *and* (3.62) *or conditions* (3.63) *and* (3.64) *in Step 4 are satisfied. Then the subsequence given by all iterates of Algorithm 3.1, which satisfy either conditions* (3.61) *and* (3.62) *or conditions* (3.63) *and* (3.64) *in Step 4, is equivalent to the iteration sequence of the linear outer approximation Algorithm 2.2.*

Proof. If conditions (3.61) and (3.62) in Step 4 are satisfied with $d_c^k = 0$ and $\eta_c^k = 0$ for some iterate (x^k, y^k), then x^k is the optimal solution of NLP(y^k) given by (3.6), see Corollary 3.1. If conditions (3.63) and (3.64) in Step 4 of Algorithm 3.1 are satisfied with $d_F^k = 0$ and $\eta^k > 0$ for some iterate (x^k, y^k), then x^k is the optimal solution of F(y^k) given by (3.7) due to Corollary 2.3 and an infeasible stationary point of NLP(y^k) according to Definition 2.10.

Furthermore, the set T_ε^k introduced in (3.43) is equivalent to the corresponding set specified by (2.71) for $d_c^k = 0$ and $\eta_c^k = 0$ in Step 4. The same holds for set S_ε^k defined in (3.44), which is equivalent to its counterpart introduced in (2.72), if $d_F^k = 0$ and $\eta^k > 0$ holds in Step 4.

Therefore Step 6 of Algorithm 3.1 is equivalent to Step 3 of Algorithm 2.2 apart from the reinitialization of some parameters in (3.67).

The reinitialization executed in (3.67) has no influence on the convergence of Yuan's trust region algorithm due to Lemma 2.2 and Theorem 2.3. The corresponding convergence properties also hold for Algorithm 3.1 due to Corollary 3.4.

Furthermore, since the sets T_ε^k and S_ε^k are equivalent to the ones defined by Fletcher and Leyffer [50] and $on_{MIQP}^k = 0$ holds $\forall k$, each integer value y^k within the considered subsequence needs to be different, see Lemma 2.5 and Lemma 2.6 or Fletcher and Leyffer [50]. Therefore the "**if** condition" in Step 6 is never satisfied.

As a consequence, the subsequence of iterates determined by Algorithm 3.1 satisfying either conditions (3.61) and (3.62) or conditions (3.63) and (3.64) in Step 4 is the

same as the iteration sequence generated by the linear outer approximation method described by Algorithm 2.2. □

Due to Corollary 3.5 we can establish convergence of Algorithm 3.1, if no mixed-integer search steps are performed and if a KKT-point of either NLP(y^k) given by (3.6) or F(y^k) given by (3.7) is obtained whenever either conditions (3.61) and (3.62) or conditions (3.63) and (3.64) are satisfied in Step 4.

Corollary 3.6. *Let* $on^k_{MIQP} = 0$, $\forall k$ *hold and assume, that either* $d^k_c = 0$ *and* $\eta^k_c = 0$ *or* $d^k_F = 0$ *and* $\eta^k > 0$ *holds, whenever conditions* (3.61) *and* (3.62) *or conditions* (3.63) *and* (3.64) *in Step 4 are satisfied. Furthermore, let Assumption 3.1 hold. Then Algorithm 3.1 solves the convex MINLP* (3.1) *or detects infeasibility.*

Proof. Due to Corollary 3.4 and Corollary 3.5 Theorem 2.4 holds in this case for Algorithm 3.1. □

3.2 Convergence Analysis

In this section we prove convergence properties of Algorithm 3.1 for the convex MINLP (3.1) subject to an optimality tolerance ε_{OA} and a feasibility tolerance ε in a finite number of iterations. An important issue, also concerning the linear outer approximation method described by Algorithm 2.2, is the finite termination of the solution process for NLP(y^k) given by (3.6) or F(y^k) given by (3.7) respectively for some $y^k \in Y$ in each iteration of Algorithm 2.2. This topic is usually neglected in existing literature but needs to be addressed, in order to show finite termination of Algorithm 3.1. To establish this task, we introduced a ε-stationary-point of NLP(y^k) and F(y^k) for fixed $y^k \in Y$ in Definition 3.1.

The subsequent lemma shows, that an integer value $y^k \in Y$ is infeasible in the master problem (3.42) for all values of $x \in X$, if conditions (3.61) and (3.62) are satisfied for some iterate (x^k, y^k). Note, that this also includes the optimal solution $\bar{x}_{y^k}$ of NLP(y^k), i.e., $(\bar{x}_{y^k}, y^k)$ does not satisfy the constraints of the master problem (3.42) if the set T^k_ε contains an iterate (x^k, y^k) satisfying conditions (3.61) and (3.62). Furthermore, (x^k, λ^k_c) is a ε-stationary-point of NLP(y^k) given by (3.6), see Definition 3.1, due to Corollary 3.2.

Lemma 3.1. *Let Assumption 3.1 hold and let* (x^k, y^k) *satisfy conditions* (3.61) *and* (3.62). *Furthermore, let* (x^k, y^k) *be included in the set* T^k_ε *introduced in* (3.43).

Then there exists no $\hat{x} \in X$, *such that* $(\hat{x}, y^k)$ *satisfies the constraints of the outer approximation master problem* (3.42).

Proof. We will prove by contradiction, that there exists no point $(\hat{x}, y^k, \hat{\eta})$, that satisfies the constraints of the outer approximation master problem (3.42), if the set T^k_ε introduced in (3.43) contains (x^k, y^k) satisfying conditions (3.61) and (3.62).

Note, that for a KKT-point $(\eta_c^k, d_c^k, (\lambda_c^k, \lambda_{\eta_c}^k))$ the KKT-conditions of $QP(x^k, y^k)$ are stated in (3.15). Farkas Lemma, see e.g., Jarre and Stoer [66], states that exactly one of the following two possibilities holds:

1.
$$\nabla_x f(x^k, y^k) + B_c^k (d_c^k)_x - [\nabla_x g(x^k, y^k)]^T \lambda_c^k \;=\; 0 \tag{3.70}$$
and $\lambda_j^k \geq 0,\ \forall j \in \mathbb{J}$.

2.
$$\nabla_x g_j(x^k, y^k)^T s \;\geq\; 0,\ \forall j \in \mathbb{J} \tag{3.71}$$
and
$$(\nabla_x f(x^k, y^k) + B_c^k (d_c^k)_x)^T s \;<\; 0, \quad \forall\, s \in \mathbb{R}^{n_c}. \tag{3.72}$$

Since (x^k, y^k) is a member of T_ε^k the subsequent conditions are required for $(\hat{x}, y^k, \hat{\eta})$, since the corresponding constraints are contained in the outer approximation master problem (3.42)

$$\hat{\eta} \;\leq\; f(x^k, y^k) - \varepsilon_{OA}, \tag{3.73}$$
$$\hat{\eta} \;\geq\; f(x^k, y^k) + \nabla_{x,y} f(x^k, y^k)^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix}, \tag{3.74}$$
$$0 \;\leq\; g_j(x^k, y^k) + \nabla_{x,y} g_j(x^k, y^k)^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix},\ \forall\, j \in \mathbb{J}, \tag{3.75}$$

where (3.73) is an upper bound on variable η introduced in (3.45).
For any constraint of $QP(x^k, y^k)$, that is active at $((d_c^k)_x, \eta_c^k)$, we get from (3.15)

$$g_j(x^k, y^k) \;=\; -\eta_c^k - \nabla_{x,y} g_j(x^k, y^k)^T d_c^k. \tag{3.76}$$

Together with (3.76), (3.75) yields

$$0 \;\leq\; -\eta_c^k - \nabla_{x,y} g_j(x^k, y^k)^T d_c^k + \nabla_{x,y} g_j(x^k, y^k)^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix} \tag{3.77}$$
$$=\; -\eta_c^k + \nabla_{x,y} g_j(x^k, y^k)^T \begin{pmatrix} \hat{x} - x^k - (d_c^k)_x \\ 0 \end{pmatrix}. \tag{3.78}$$

As a consequence, we obtain

$$0 \overset{(3.15)}{\leq} \eta_c^k \;\leq\; \nabla_x g_j(x^k, y^k)^T \left(\hat{x} - x^k - (d_c^k)_x\right). \tag{3.79}$$

Since the KKT conditions of $QP(x^k, y^k)$ are satisfied, (3.70) holds with $(\lambda_c^k)_j \geq 0,\ \forall j \in \mathbb{J}$. As (3.71) is also satisfied as shown in (3.79), Farkas lemma (3.72) yields

$$(\nabla_x f(x^k, y^k) + B_c^k (d_c^k)_x)^T s \;\geq\; 0. \tag{3.80}$$

Let s be given by

$$s := \hat{x} - x^k - (d_c^k)_x. \tag{3.81}$$

then

$$(\nabla_x f(x^k, y^k) + B_c^k (d_c^k)_x)^T \left(\hat{x} - x^k - (d_c^k)_x\right) \geq 0, \tag{3.82}$$

holds, which gives

$$\nabla_x f(x^k, y^k)^T \left(\hat{x} - x^k\right) \geq \nabla_x f(x^k, y^k)^T (d_c^k)_x - (d_c^k)_x^T B_c^k s. \tag{3.83}$$

Exploiting conditions (3.61) and (3.62), which hold by assumption together with (3.83), condition (3.74) yields

$$\begin{aligned}
\hat{\eta} &\geq f(x^k, y^k) + \nabla_{x,y} f(x^k, y^k)^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix} \\
&\overset{(3.83)}{\geq} f(x^k, y^k) + \nabla_x f(x^k, y^k)^T (d_c^k)_x - (d_c^k)_x^T B_c^k \left(\hat{x} - x^k - (d_c^k)_x\right) \qquad (3.84) \\
&\geq f(x^k, y^k) + \nabla_x f(x^k, y^k)^T (d_c^k)_x + (d_c^k)_x^T B_c^k (d_c^k)_x - (d_c^k)_x^T B_c^k \left(\hat{x} - x^k\right) \\
&\geq f(x^k, y^k) - |\nabla_x f(x^k, y^k)^T (d_c^k)_x| - |(d_c^k)_x^T B_c^k (d_c^k)_x| - |(d_c^k)_x^T B_c^k \left(\hat{x} - x^k\right)| \\
&\geq f(x^k, y^k) - \|\nabla_x f(x^k, y^k)\|_2 \|(d_c^k)_x\|_2 - \|(d_c^k)_x^T B_c^k\|_2 \|(d_c^k)_x\|_2 \\
&\quad - \|(d_c^k)_x^T B_c^k\|_2 \| \left(\hat{x} - x^k\right) \|_2 \\
&\overset{(3.5)}{\geq} f(x^k, y^k) - \|\nabla_x f(x^k, y^k)\|_2 \|(d_c^k)_x\|_2 - \|(d_c^k)_x^T B_c^k\|_2 \|(d_c^k)_x\|_2 - \|(d_c^k)_x^T B_c^k\|_2 M_x \\
&\overset{(3.62)}{>} f(x^k, y^k) - \varepsilon_{OA}.
\end{aligned}$$

This contradicts condition (3.73) and shows, that no $(\hat{x}, y^k, \hat{\eta})$ exists, that satisfies the constraints of the outer approximation master problem (3.42), if a ε-stationary point of NLP(y^k) is included in T^k. □

Note, that Lemma 3.1 applies to Algorithm 3.1 in Step 4, if conditions (3.61) and (3.62) are satisfied for an iterate (x^k, y^k).

To be able to show finite termination of Algorithm 3.1, the same has to be true, if conditions (3.63) and (3.64) are satisfied for the current iterate (x^k, y^k). This is ensured by the subsequent lemma. Note, that, $(x^k, \eta^k, (\lambda_F^k, \lambda_\eta^k))$ is a ε-stationary point of $F(y^k)$, due to Corollary 3.3.

Lemma 3.2. *Let Assumption 3.1 hold and let* (x^k, y^k) *satisfy conditions* (3.63) *and* (3.64). *Furthermore, let* (x^k, y^k) *be included in the set* S_ε^k *introduced in* (3.44).

Then there exists no $\hat{x} \in X$, *such that* $(\hat{x}, y^k)$ *satisfies the constraints of the outer approximation master problem* (3.42).

Proof. We will prove by contradiction, that there exists no point $(\hat{x}, y^k, \hat{\eta})$, that satisfies the constraints of the outer approximation master problem (3.42), if the set S_ε^k introduced in (3.44) contains (x^k, y^k) satisfying conditions (3.63) and (3.64).

Since (x^k, y^k) is included in the set S_ε^k, the subsequent conditions are required for $(\hat{x}, y^k, \hat{\eta}) \in X \times Y \times \mathbb{R}$ that is feasible for the outer approximation master problem (3.42), since it contains the corresponding constraints:

$$g_j(x^k, y^k) + \nabla_{x,y} g_j(x^k, y^k)^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix} \geq 0, \ \forall j \in \mathbb{J}. \tag{3.85}$$

The constraints $g_j, \ j \in \mathbb{J}$ can be divided into two sets according to the value of the constraint violation of LPF(x^k, y^k) at the solution $((d_F^k)_x, \eta^k)$. We define

$$\mathbb{W} := \left\{ j \in \mathbb{J} : g_j(x^k, y^k) + \nabla_x g_j(x^k, y^k)^T (d_F^k)_x = -\eta^k \right\} \tag{3.86}$$

and

$$\mathbb{W}^\perp := \mathbb{J} \backslash \mathbb{W}. \tag{3.87}$$

For each $j \in \mathbb{W}^\perp$ the corresponding Lagrangian multiplier $(\lambda_F^k)_j \in \mathbb{R}_+$ associated with the solution $((d_F^k)_x, \eta^k)$ is zero, i.e.,

$$(\lambda_F^k)_j = 0, \ \forall j \in \mathbb{W}^\perp \tag{3.88}$$

holds. Furthermore $\lambda_\eta^k = 0$ holds due to (3.64). The KKT-conditions of LPF(x^k, y^k) stated in (2.98) yield

$$1 = \sum_{j \in \mathbb{W}} (\lambda_F^k)_j, \tag{3.89}$$

$$0 = \sum_{j \in \mathbb{W}} (\lambda_F^k)_j \nabla_x g_j(x^k, y^k). \tag{3.90}$$

After multiplying all inequalities (3.85) with $(\lambda_F^k)_j, \ j \in \mathbb{J}$ and summing up, we get

$$\sum_{j \in \mathbb{W}} (\lambda_F^k)_j g_j(x^k, y^k) + \left(\sum_{j \in \mathbb{W}} (\lambda_F^k)_j \nabla_{x,y} g_j(x^k, y^k) \right)^T \begin{pmatrix} \hat{x} - x^k \\ 0 \end{pmatrix} \geq 0, \tag{3.91}$$

since $(\lambda_F^k)_j = 0, \ \forall j \notin \mathbb{W}$. Exploiting equation (3.90) yields

$$\sum_{j \in \mathbb{W}} (\lambda_F^k)_j g_j(x^k, y^k) \geq 0. \tag{3.92}$$

Since $j \in \mathbb{W}$ holds, we get a contradiction due to

$$\begin{aligned} 0 &\leq \sum_{j\in\mathbb{W}} (\lambda_F^k)_j g_j(x^k, y^k) \\ &\overset{(2.101)}{=} \sum_{j\in\mathbb{W}} (\lambda_F^k)_j (-\nabla_x g_j(x^k, y^k)^T (d_F^k)_x - \eta^k) \\ &\leq \sum_{j\in\mathbb{W}} (\lambda_F^k)_j (|\nabla_x g_j(x^k, y^k)^T (d_F^k)_x| - \eta^k) \\ &\leq \sum_{j\in\mathbb{W}} (\lambda_F^k)_j (\|\nabla_x g_j(x^k, y^k)\|_2 \|(d_F^k)_x\|_2 - \eta^k) \\ &\overset{(3.63),(3.64)}{<} \sum_{j\in\mathbb{W}} (\lambda_F^k)_j (\varepsilon - \varepsilon) \\ &= 0. \end{aligned} \tag{3.93}$$

As a consequence, there exists no $(\hat{x}, y^k, \hat{\eta}) \in X \times Y \times \mathbb{R}$, that satisfies the conditions (3.85) posed by the outer approximation master problem (3.42), if S_ε^k contains an iterate (x^k, y^k) satisfying (3.63) and (3.64). □

Note, that Lemma 3.2 applies to Algorithm 3.1 in Step 4, if the current iterate (x^k, y^k) satisfies (3.63) and (3.64).

Lemmata 3.1 and 3.2 ensure, that the condition requiring either $d_c^k = 0$ and $\eta_c^k = 0$ or $d_F^k = 0$ and $\eta^k > 0$ whenever conditions (3.61) and (3.62) or conditions (3.63) and (3.64) are satisfied in Step 4, is not needed anymore in Corollary 3.6. Furthermore, we can establish below, that Algorithm 3.1 terminates after a finite number of iterations. If $f^* < \infty$ holds at termination, then the algorithm found a ε-solution (x^*, y^*) defined as follows.

Definition 3.3. *A point $(x^*, y^*) \in X \times Y$ is a ε-solution of the convex MINLP (3.1), if conditions*

$$\|g(x^*, y^*)^-\|_\infty \leq \varepsilon \tag{3.94}$$

and

$$f(x^*, y^*) \leq \min_{\hat{y}\in V} \max_{\hat{x}} \{f(\hat{x}, \hat{y}) : (\hat{x}, \hat{\lambda}_c) \text{ is } \varepsilon\text{-stationary-point of NLP}(\hat{y})\} + \varepsilon_{OA} \tag{3.95}$$

are satisfied, where ε_{OA} is the chosen optimality tolerance and ε is the chosen feasibility tolerance and $\hat{\lambda}_c$ is the Lagrangian multiplier obtained by solving QP($\hat{x}, \hat{y}$).

Note, that the maximum in (3.95) can be removed, if optimal solutions instead of ε-stationary-points of NLP(y^k), or F(y^k) respectively, are obtained. Then the ε-solution defined above corresponds to the global solution of a convex MINLP (3.1) subject to the optimality tolerance ε_{OA}.

Note, that there exist infinitely many ε-solutions of the convex MINLP (3.1) for any values of ε_{OA} and ε, as long as $V \neq \emptyset$. There reason is, that there exist infinitely many ε-stationary points for any NLP(y^k) with $y^k \in V$ for any value of $\varepsilon > 0$.

To carry out the convergence analysis, we have to introduce an additional assumption. It is necessary to exclude infeasible stationary points, that possess a constraint violation smaller than the feasibility tolerance ε. By choosing the feasibility tolerance sufficiently small, this additional assumption is no severe limitation.

Assumption 3.2. *Without loss of generality, we assume, that the solution $\bar{x}_{y^k}$ of F(y^k) possesses a maximal constraint violation larger than the feasibility tolerance ε for each $y^k \in Y \backslash V$, i.e.,*

$$\|g(\bar{x}_{y^k}, y^k)^-\|_\infty \;>\; \varepsilon \tag{3.96}$$

holds for each $y^k \in Y \backslash V$.

As a consequence, Corollary 3.6 can be replaced by the subsequent corollary.

Corollary 3.7. *Let $on^k_{MIQP} = 0$ hold $\forall k$. Furthermore, let Assumption 3.1 and Assumption 3.2 hold. Then Algorithm 3.1 terminates after a finite number of iterations either at a ε-solution of the convex MINLP* (3.1) *according to Definition 3.3 or detects, that no feasible point exists, i.e., $V = \emptyset$, where V is introduced by* (3.8).

Proof. Consider some fixed integer value $y^{\bar{k}}$. Let $y^{\bar{k}}$ be in V, then $(\hat{x}, y^{\bar{k}}) \in X \times V$ is infeasible for all values of $\hat{x} \in X$ in the master problem (3.42), if the set T^k_ε contains a ε-stationary point of NLP($y^{\bar{k}}$), see Lemma 3.1. Alternatively let $y^{\bar{k}}$ be in $Y \backslash V$ instead, then $(\hat{x}, y^{\bar{k}}) \in X \times Y \backslash V$ is infeasible for all values of $\hat{x} \in X$ in the master problem (3.42), if the set S^k_ε contains a ε-stationary point of F($y^{\bar{k}}$), see Lemma 3.2

As a consequence $y^{\bar{k}} \in Y$ can not be considered twice, if a ε-stationary point of NLP($y^{\bar{k}}$) is contained in the set T^k_ε or respectively a ε-stationary point of F($y^{\bar{k}}$) is contained in the set S^k_ε. In combination with the finiteness of Y this ensures, that Algorithm 3.1 terminates after a finite number of iterations, if a ε-stationary point of either NLP($y^{\bar{k}}$) or F($y^{\bar{k}}$) is obtained after a finite number of iterations for all $y^{\bar{k}} \in Y$.

Therefore, we prove, that the execution of the Steps 2, 4, 5 and 7 of Algorithm 3.1 yields a ε-stationary point of either NLP($y^{\bar{k}}$) or F($y^{\bar{k}}$) after a finite number of iterations starting at an arbitrary iterate $(x^{\bar{k}}, y^{\bar{k}})$.

As long as neither a ε-stationary point of NLP($y^{\bar{k}}$) nor F($y^{\bar{k}}$) is obtained, $y^k = y^{\bar{k}}$ holds for the iteration sequence (x^k, y^k) with $k \geq \bar{k}$. Due to Corollary 2.5, and Corollary 3.4 the iteration sequence (x^k, y^k) with $k \geq \bar{k}$ obtained by executing Steps 2, 4, 5 and 7 of Algorithm 3.1 converges towards a stationary point $\bar{x}_{y^{\bar{k}}}$ of NLP($y^{\bar{k}}$), if $y^{\bar{k}} \in V$ or alternatively F($y^{\bar{k}}$), if $y^{\bar{k}} \in Y \backslash V$. Since the KKT-conditions of either NLP($y^{\bar{k}}$) or F($y^{\bar{k}}$) are satisfied for the accumulation point $(\bar{x}_{y^{\bar{k}}}, y^{\bar{k}})$, the conditions (3.17) or (3.18) defining a ε-stationary point of NLP($y^{\bar{k}}$) or alternatively F($y^{\bar{k}}$) are satisfied after a finite number of iterations for an arbitrary small fixed value of $\varepsilon > 0$. Therefore, we proved the finite termination of Algorithm 3.1.

Now, we prove by contradiction, that no feasible point exists, i.e., $\nexists\ (\hat{x}, \hat{y}) \in X \times Y$ with $g_j(\hat{x}, \hat{y}) \geq 0,\ \forall j \in \mathbb{J}$, if Algorithm 3.1 terminates with $f^* = \infty$.

Assume, that $\exists\ (\hat{x},\hat{y}) \in X \times Y$ with $g_j(\hat{x},\hat{y}) \geq 0,\ \forall j \in \mathbb{J}$. Since $g_j(x,y)$ is concave for all $j \in \mathbb{J}$, $(\hat{x},\hat{y})$ cannot be cut off in the master problem (3.42) by any linearization of some $g_j(x,y),\ j \in \mathbb{J}$. Since $f^* = \infty$ holds, no linearization of $f(x,y)$ is contained in the master problem. As a consequence, $(\hat{x},\hat{y})$ is still feasible for the master problem, which contradicts to the termination of Algorithm 3.1.

Moreover, we prove by contradiction, that (x^*,y^*) is a ε-solution according to Definition 3.3, if Algorithm 3.1 returns with $f^* < \infty$. Note first, that (x^*,y^*) is feasible subject to the feasibility tolerance ε, due to the update rule of the best known solution in Algorithm 3.1.

Assume, that there exists a $\hat{y} \in V$ with $\hat{f} + \varepsilon_{OA} \leq f^*$ at termination, where $\hat{f}$ is the maximal function value of all ε-stationary-points of NLP($\hat{y}$), i.e.,

$$\hat{f} \ := \ \max_{\hat{x}} \{ f(\hat{x},\hat{y}) : (\hat{x},\hat{\lambda}_c) \text{ is } \varepsilon\text{-stationary point of NLP}(\hat{y}) \},$$

where $\hat{\lambda}_c$ is the Lagrangian multiplier obtained by solving QP($\hat{x},\hat{y}$). Since $\hat{y} \in V$, there exists a feasible ε-stationary point denoted by $\hat{x}_F$ of NLP($\hat{y}$), i.e.,

$$\hat{x}_F \ := \ \arg\max_{\hat{x}} \left\{ \begin{array}{l} f(\hat{x},\hat{y}) : (\hat{x},\hat{\lambda}_c) \text{ is } \varepsilon\text{-stationary point of NLP}(\hat{y}) \text{ and} \\ g_j(\hat{x},\hat{y}) \geq 0, \forall j \in \mathbb{J} \end{array} \right\}.$$

Denote the final iterate by k^*. Since $\hat{f}+\varepsilon_{OA} < f^*$, $T_\varepsilon^{k^*}$ contains no ε-stationary-point of NLP($\hat{y}$), otherwise $f^* \leq \hat{f}$ holds. Furthermore, $f(\hat{x}_F,\hat{y}) \leq \hat{f}$ and $g_j(\hat{x}_F,\hat{y}) \geq 0,\ \forall j \in \mathbb{J}$ holds for $(\hat{x}_F,\hat{y})$ by construction.

For $(\hat{x}_F,\hat{y})$ the subsequent conditions are derived from the linearizations of the constraints $g_j(x,y),\ j \in \mathbb{J}$, which are contained in the master problem (3.42):

$$g_j(x^i,y^i) + \nabla_{x,y} g_j(x^i,y^i)^T \begin{pmatrix} \hat{x}_F - x^i \\ \hat{y} - y^i \end{pmatrix} \ \geq \ 0,\ \forall j \in \mathbb{J} \tag{3.97}$$

with $(x^i,y^i) \in T_\varepsilon^{k^*}$ or $(x^i,y^i) \in S_\varepsilon^{k^*}$. They are satisfied, since $g_j(x,y)$ is concave for all $j \in \mathbb{J}$ and $g_j(\hat{x}_F,\hat{y}) \geq 0,\ \forall j \in \mathbb{J}$ holds.

Furthermore, for $(\hat{x}_F,\hat{y})$ also the subsequent conditions are derived from the linearizations of the objective function $f(x,y),\ j \in \mathbb{J}$, which are contained in the master problem (3.42):

$$f(x^i,y^i) + \nabla_{x,y} f(x^i,y^i)^T \begin{pmatrix} \hat{x}_F - x^i \\ \hat{y} - y^i \end{pmatrix} \ \leq \ \eta, \tag{3.98}$$

$$\eta \ \leq \ f^* - \varepsilon_{OA} \tag{3.99}$$

with $(x^i,y^i) \in T_\varepsilon^{k^*}$. They are also satisfied even in combination with the upper bound on η, since we obtain

$$f(x^i,y^i) + \nabla_{x,y} f(x^i,y^i)^T \begin{pmatrix} \hat{x}_F - x^i \\ \hat{y} - y^i \end{pmatrix} \ \leq \ f(\hat{x}_F,\hat{y}) \ < \ f^* - \varepsilon_{OA} \tag{3.100}$$

by exploiting the convexity of the objective function and $f(\hat{x}_F, \hat{y}) < f^* - \varepsilon_{OA}$. As a consequence $(\hat{x}_F, \hat{y})$ is feasible in the master problem MILP$(T_\varepsilon^{k^*}, S_\varepsilon^{k^*}, f^*, \varepsilon_{OA})$, which contradicts to the termination of the algorithm and proves the corollary.

□

Up to now, we proved, that Algorithm 3.1 terminates after a finite number of iterations at a ε-solution according to Definition 3.3 or it detects that no feasible point exists, if no mixed-integer search steps determined by the solution of mixed-integer problem (3.48) are performed, i.e., $on_{MIQP}^k = 0, \ \forall k$. To show, that the convergence properties of Algorithm 3.1 are maintained, if mixed-integer search steps are carried out, we consider two cases separately.

First we assume, that the execution of the Steps 2, 3, 4, 5 and 7 of Algorithm 3.1 always determines an iterate $(x^{\bar{k}}, y^{\bar{k}})$ after a finite number of iterations, which satisfies either conditions (3.61) and (3.62) or conditions (3.63) and (3.64) in Step 4.

Lemma 3.3. *Let Assumption 3.1 and Assumption 3.2 hold. Furthermore, let the execution of the Steps 2, 3, 4, 5 and 7 of Algorithm 3.1 always determine an iterate* $(x^{\bar{k}}, y^{\bar{k}})$ *after a finite number of iterations, that satisfies either conditions* (3.61) *and* (3.62) *or conditions* (3.63) *and* (3.64) *in Step 4 of Algorithm 3.1 for an arbitrary* $y^{\bar{k}} \in Y$.

Then Algorithm 3.1 terminates after a finite number of iterations at a ε-solution according to Definition 3.3 or it detects that no feasible point exists.

Proof. It is sufficient to prove the finite termination of Algorithm 3.1. The proofs given in Corollary 3.7, which ensure that a ε-solution is obtained, if $f^* < \infty$ holds and that no feasible point exists, if $f^* = \infty$ holds at termination, remain valid.

By assumption, the execution of the Steps 2, 3, 4, 5 and 7 of Algorithm 3.1 always determine an iterate $(x^{\bar{k}}, y^{\bar{k}})$ after a finite number of iterations, that satisfies conditions (3.61) and (3.62) or conditions (3.63) and (3.64) for some $y^{\bar{k}} \in Y$. Since the set Y is finite, an infinite number of iterations must be caused by infinite repetition of Step 6 of Algorithm 3.1. We show by contradiction, that this is not possible:

Assume, that Step 6 of Algorithm 3.1 is repeated an infinite number of times. Due to the finiteness of the set Y, there exists a subset $\hat{Y} \subset Y$ of integer values, that are feasible for the master problem at each execution of Step 6. Now we run Algorithm 3.1 until the solution of the master problem (3.42) possesses the integer values of some $\hat{y} \in \hat{Y}$ for the second time. Then the integer values are fixed to $\hat{y}$ by setting $on_{MIQP}^k = 0$ in Step 6 of Algorithm 3.1. In this case an iterate $(x^{\bar{k}}, \hat{y})$ satisfying conditions (3.61) and (3.62) or conditions (3.63) and (3.64) is determined by executing Steps 2, 3, 4, 5 and 7 by assumption, see also Corollary 3.4, Lemma 2.2 and Theorem 2.3. As a consequence of Lemma 3.1 or Lemma 3.2, $(x, \hat{y})$ is then infeasible in the master problem for all $x \in X$ whenever Step 6 is executed. This contradicts the assumption, that $\hat{y}$ is in the set $\hat{Y}$ and proves the lemma. □

We still have to prove that the execution of the Steps 2, 3, 4, 5 and 7 of Algorithm 3.1 converges towards a stationary point according to Definition 2.12 or an infeasible stationary point according to Definition 2.10 of problem NLP($y^{\bar{k}}$) given by (3.6) for some $y^{\bar{k}} \in Y$. This is established by the following lemma. The convergence ensures that either conditions (3.61) and (3.62) or conditions (3.63) and (3.64) in Step 4 of Algorithm 3.1 are satisfied after a finite number of iterations, see Corollary 3.7.

Lemma 3.4. *If Assumption 3.1 and Assumption 3.2 hold, then the successive execution of the Steps 2, 3, 4, 5 and 7 of Algorithm 3.1 yields an iterate* $(x^{\bar{k}}, y^{\bar{k}})$ *satisfying conditions* (3.61) *and* (3.62) *or conditions* (3.63) *and* (3.64) *after a finite number of iterations for some integer value* $y^{\bar{k}} \in Y$*. I.e., Algorithm 3.1 reaches Step 6.*

Proof. First we prove convergence, for some special cases, where the number of iterations, in which an improving mixed-integer search direction according to Definition 3.2 exists, is finite. The open case, where the number of iterations, in which an improving mixed-integer search direction exists, is infinite, is proved by adapting the proof of Yuan [112] of Theorem 2.3 based on an appropriate redefinition of the iteration sequence of Algorithm 3.1.

Note, that the successive execution of the Steps 2, 3, 4, 5 and 7 yields the same iteration sequence $\{(x^k, y^{\bar{k}})\}$ as the trust region Algorithm 2.1 of Yuan applied to solve NLP($y^{\bar{k}}$) given by (3.6), if no mixed-integer steps are performed, see Corollary 3.4 and $y^k = y^{\bar{k}}$ holds for all $k > \bar{k}$ until the conditions (3.61) and (3.62) or conditions (3.63) and (3.64) are satisfied. This is the case, if no improving mixed-integer search direction according to Definition 3.2 exists or if $on^k_{MIQP} = 0$ holds in iteration k.

The successive execution of the Steps 2, 3, 4, 5 and 7 can either be associated with a bounded or an unbounded sequence of penalty parameters $\{\sigma^k\}$. If the sequence of penalty parameters $\{\sigma^k\}$ is unbounded, then there exists an iteration $\hat{k}$, where the value of the penalty parameter $\sigma^{\hat{k}}$ is larger than the threshold $\bar{\sigma}$, specified in Algorithm 3.1. As a consequence, we skip the solution of MIQP(x^k, y^k) given by (3.50) for all $k \geq \hat{k}$ until either the conditions (3.61) and (3.62) or conditions (3.63) and (3.64) hold. In this case no improving mixed-integer search directions according to Definition 3.2 can be obtained. Convergence is then proved by Corollary 3.4, since the sequence of penalty parameter is monotone increasing until Step 6 is reached.

It remains to prove convergence, if the sequence of penalty parameters $\{\sigma^k\}$ is bounded by $\bar{\sigma}$. Without loss of generality, we consider iteration $\bar{k}$ with $\sigma^{\bar{k}} = \bar{\sigma}$, i.e., the penalty parameter remains constant for all successive iterations until either conditions (3.61) and (3.62) or conditions (3.63) and (3.64) are satisfied in Step 4.

In principle, the successive execution of the Steps 2, 3, 4, 5 and 7 can either contain a finite or an infinite number of iterations, where an improving mixed-integer search direction according to Definition 3.2 exists. First we assume, that the number of iterations, in which an improving mixed-integer search direction according to Definition 3.2 exists, is finite. Then we know that the iteration sequence $\{(x^k, y^k)\},\ k \geq \tilde{k}$ converges towards a stationary point of NLP($y^{\bar{k}}$) specified in Definition 2.12 due to

Theorem 2.3, see also Corollary 3.4, where $\tilde{k} \geq \bar{k}$ denotes the last iteration, in which an improving mixed-integer search direction according to Definition 3.2 exists.

Now we assume, that there are infinitely many iterations, in which an improving mixed-integer search direction according to Definition 3.2 exists. Since the set Y is finite, there exists a subset $\hat{Y} \subset Y$ of integer values, such that every $\hat{y} \in \hat{Y}$ occurs infinitely many times in the iteration sequence $\{(x^k, y^k)\}$ determined by the Steps 2, 3, 4, 5 and 7 of Algorithm 3.1.

We consider an arbitrary $\hat{y} \in \hat{Y}$. We will prove by contradiction in the same way as Yuan [112] or Jarre and Stoer [66], that the iteration sequence $\{(x^k, y^k)\}$ determined by the Steps 2, 3, 4, 5 and 7 of Algorithm 3.1 converges towards a stationary point $\bar{x}_{\hat{y}}$ of NLP($\hat{y}$), see Definition 2.12. Therefore we assume, that such stationary point $\bar{x}_{\hat{y}}$ is not an accumulation point of the iteration sequence $\{(x^k, y^k)\}$.

Furthermore, we assume without loss of generality, that an improving mixed-integer search direction $d_i^{\bar{k}}$ according to Definition 3.2 exists in iteration $\bar{k}$ with $y^{\bar{k}} = \hat{y}$. Since $\hat{y} \in \hat{Y}$ holds, we know, that the integer value $\hat{y}$, reoccurs in some iteration denoted by $\hat{k}$, i.e., $\hat{y} = y^{\bar{k}} = y^{\hat{k}}$. As a consequence, we can define the continuous search step $\tilde{d}^k$, which subsumes all search steps inbetween two successive iterations with integer values $\hat{y}$, e.g., $\tilde{d}^{\bar{k}}$ given by

$$\tilde{d}^{\bar{k}} := \begin{pmatrix} x^{\hat{k}} - x^{\bar{k}} \\ 0 \end{pmatrix}. \tag{3.101}$$

In the remainder, we consider the corresponding iteration sequence $\{(\hat{x}^k, \hat{y})\}$, $k \geq \bar{k}$, with

$$\hat{x}^{k+1} := \begin{cases} \hat{x}^k + d_c^k, & \text{if no improving mixed-integer search direction exists in iteration } k \\ & \text{and } r_c^k > 0 \text{ holds,} \\ \hat{x}^k, & \text{if no improving mixed-integer search direction exists in iteration } k \\ & \text{and } r_c^k \leq 0 \text{ holds,} \\ \hat{x}^k + \tilde{d}^k, & \text{if an improving mixed-integer search direction exists in iteration } k. \end{cases}$$

Since the sequence of penalty parameters is bounded, we know that

$$\hat{\Phi}_c^k(0) - \hat{\Phi}_c^k(d_c^k) \;>\; \delta^k \bar{\sigma} \min\{\Delta_c^k, \|g(\hat{x}^k, \hat{y})^-\|_\infty\} \tag{3.102}$$

is satisfied for all $k \geq \bar{k}$, where $\delta^k \in \mathbb{R}_+$ is the parameter of the penalty parameter update of Algorithm 3.1 and $\hat{\Phi}_c^k$ is defined by

$$\begin{aligned} \hat{\Phi}_c^k((d_c)_x) \;:=\; & \nabla_x f(\hat{x}^k, y^k)^T (d_c)_x + \frac{1}{2} (d_c)_x^T B_c^k (d_c)_x \\ + \; & \sigma^k \|(g(\hat{x}^k, y^k) + [\nabla_x g(\hat{x}^k, y^k)]^T (d_c)_x)^-\|_\infty, \end{aligned} \tag{3.103}$$

see also (3.11) and note, that $B_c^k \in \mathbb{R}^{n_c \times n_c}$ is the upper left sub-matrix of the $n \times n$-matrix B^k, which is symmetric and positive definite. Otherwise the penalty parameter σ^k does not remain constant due to the penalty parameter update.

For fixed $\hat{y} \in Y$ let Ω be defined as the closure of the set of feasible iterates given by

$$\Omega \;:=\; \{(\tilde{x},\hat{y}) \in \overline{\{(\hat{x}^k,\hat{y}) | k > \bar{k}\}} : \|g(\tilde{x},\hat{y})^-\|_\infty = 0\}. \tag{3.104}$$

For $(\tilde{x},\hat{y}) \in \Omega$ we define

$$\bar{\Phi}(d_x) := \nabla_x f(\tilde{x},\hat{y})^T d_x + \frac{1}{2} M_B \|d_x\|_2^2 + \bar{\sigma}\|(g(\tilde{x},\hat{y}) + [\nabla_x g(\tilde{x},\hat{y})]^T d_x)^-\|_\infty, \tag{3.105}$$

for $d_x \in \mathbb{R}^{n_c}$ with $d := \begin{pmatrix} d_x \\ 0 \end{pmatrix}$, $d \in R^n$. M_B is a positive constant satisfying

$$\|B_c^k\|_2 \;\leq\; M_B, \;\forall k, \tag{3.106}$$

which exists due to Assumption 3.1.

Since we assume, that no stationary point of NLP($\hat{y}$) is contained in the set Ω, there exists a constant $\bar{\rho} > 0$, such that

$$\min_{\|d_x\|_\infty \leq 1} (\bar{\Phi}(d_x) - \bar{\Phi}(0)) = -\bar{\rho}. \tag{3.107}$$

As Ω is a compact set, $\bar{\rho}$ is independent of $(\tilde{x},\hat{y}) \in \Omega$. For $(\hat{x}^k,\hat{y})$ we define

$$\Psi^k(d_x) := \nabla_x f(\hat{x}^k,\hat{y})^T d_x + \frac{1}{2} M_B \|d_x\|_2^2 + \bar{\sigma}\|(g(\hat{x}^k,\hat{y}) + [\nabla_x g(\hat{x}^k,\hat{y})]^T d_x)^-\|_\infty. \tag{3.108}$$

Since $\hat{\Phi}_c^k(d_x) \leq \Psi^k(d_x)$ holds, we obtain

$$\begin{aligned} \min_{\|d_x\|_\infty \leq \Delta_c^k} (\hat{\Phi}_c^k(d_x) - \hat{\Phi}_c^k(0)) \;&\leq\; \min_{\|d_x\|_\infty \leq \Delta_c^k} (\Psi^k(d_x) - \Psi^k(0)) \\ &\leq\; \min_{\|d_x\|_\infty \leq 1} (\Psi^k(d_x) - \Psi^k(0)) \cdot \min\{\Delta_c^k, 1\} \end{aligned} \tag{3.109}$$

by exploiting $\hat{\Phi}_c^k(0) = \Psi^k(0)$ and the convexity of $\Psi(\lambda d_x)$ with respect to λ, see Jarre and Stoer [66] for further details.

Due to the compactness of Ω and the continuity of $\nabla_{x,y} f(x,y)$ and $\nabla_{x,y} g(x,y)$, there exists a $\tilde{\rho} > 0$ and a $(\tilde{x},\hat{y}) \in \Omega$, such that for all $(\hat{x}^k,\hat{y})$ with $\operatorname{dist}((\hat{x}^k,\hat{y}),\Omega) \leq \tilde{\rho}$,

$$|\Psi^k(d_x) - \bar{\Phi}(d_x)| \;\leq\; \frac{\bar{\rho}}{2}, \;\forall\; d_x \in \mathbb{R}^{n_c}, \text{ with } \|d_x\|_\infty \leq 1, \tag{3.110}$$

with

$$\operatorname{dist}(x, Z) := \min_{z \in Z}\{\|x - z\|_2\} \tag{3.111}$$

holds.

If $\operatorname{dist}((\hat{x}^k,\hat{y}),\Omega) \leq \tilde{\rho}$ holds, (3.110) yields

$$\begin{aligned} \min_{\|d_x\|_\infty \leq \Delta_c^k} (\hat{\Phi}_c^k(d_x) - \hat{\Phi}_c^k(0)) \;&\leq\; \min_{\|d_x\|_\infty \leq 1} (\Psi^k(d_x) - \Psi^k(0)) \cdot \min\{\Delta_c^k, 1\} \\ &\leq\; \min_{\|d_x\|_\infty \leq 1} (\bar{\Phi}(d_x) - \bar{\Phi}(0) + \tfrac{\bar{\rho}}{2}) \cdot \min\{\Delta_c^k, 1\} \\ &\leq\; -\tfrac{\bar{\rho}}{2} \cdot \min\{\Delta_c^k, 1\}. \end{aligned} \tag{3.112}$$

Due to the boundedness of $\{\Delta_c^k\}$, we obtain

$$\min_{\|d_x\|_\infty \le \Delta_c^k} (\hat{\Phi}_c^k(d_x) - \hat{\Phi}_c^k(0)) \;\le\; -\bar{\delta}\Delta_c^k, \tag{3.113}$$

for some small $\bar{\delta} > 0$.

For $\operatorname{dist}((\hat{x}^k, \hat{y}), \Omega) > \tilde{\rho}$, the definition of Ω ensures, that $\|g(\hat{x}^k, \hat{y})^-\|_\infty \ge \hat{\rho}$ holds for some $\hat{\rho} > 0$. Furthermore, we obtain from (3.102)

$$\begin{aligned} \hat{\Phi}_c^k(d_x) - \hat{\Phi}_c^k(0) &\le -\delta\bar{\sigma}\min\{\Delta_c^k, \|g(\hat{x}^k, \hat{y})^-\|_\infty\} \\ &\le -\tilde{\delta}\Delta_c^k, \end{aligned} \tag{3.114}$$

with $\tilde{\delta} > 0$.

As a consequence of (3.113) and (3.114)

$$\hat{\Phi}_c^k(0) - \hat{\Phi}_c^k(d_x) \;\ge\; \hat{\delta}\Delta_c^k, \tag{3.115}$$

holds for all k for some $\hat{\delta} > 0$.

Let the set of iterations with $(\hat{x}^{k+1}, \hat{y}) = (\hat{x}^k, \hat{y}) + \tilde{d}^k$ be denoted by K_1. Furthermore denote the set of iterations with $(\hat{x}^{k+1}, \hat{y}) = (\hat{x}^k, \hat{y}) + d_c^k$ with $r_c^k \ge 0.1$ in (3.53) by K_2. The set $K := K_1 \cup K_2$ is considered to be the set of successful iterations yielding a reduction with $r_c^k \ge 0.1$ in the merit function. Due to the boundedness of $P_{\bar{\sigma}}$ introduced in Definition 2.9, we obtain

$$\begin{aligned} \infty &> \sum_{k=1}^{\infty} \left(P_{\bar{\sigma}}(\hat{x}^k, \hat{y}) - P_{\bar{\sigma}}(\hat{x}^{k+1}, \hat{y})\right) \\ &\ge \sum_{k\in K_1} \left(P_{\bar{\sigma}}(\hat{x}^k, \hat{y}) - P_{\bar{\sigma}}(\hat{x}^{k+1}, \hat{y})\right) + \sum_{k\in K_2} \left(P_{\bar{\sigma}}(\hat{x}^k, \hat{y}) - P_{\bar{\sigma}}(\hat{x}^{k+1}, \hat{y})\right) \\ &\ge 0.1 \sum_{k\in K_1} \left(\hat{\Phi}_c^k(0) - \hat{\Phi}_c^k((d_c^k)_x)\right) + 0.1 \sum_{k\in K_2} \left(\hat{\Phi}_c^k(0) - \hat{\Phi}_c^k((d_c^k)_x)\right) \\ &\ge 0.1 \sum_{k\in K} \left(\hat{\Phi}_c^k(0) - \hat{\Phi}_c^k((d_c^k)_x)\right) \\ &\ge 0.1\hat{\delta} \sum_{k\in K} \Delta_c^k \end{aligned} \tag{3.116}$$

by exploiting (3.115), Definition 3.2 and

$$P_{\bar{\sigma}}(\hat{x}^{k+1}, \hat{y}) \;\le\; P_{\bar{\sigma}}((\hat{x}^k, \hat{y}) + d_i^k), \quad \forall k \in K_1,$$

where d_i^k is the improving mixed-integer search direction specified in Definition 3.2. Therefore $\sum_{k\in K} \Delta_c^k < \infty$ holds, which implies $\sum_{k=1}^{\infty} \Delta_c^k < \infty$ yielding $\{\Delta_c^k\} \to 0$. The continuity assumption gives

$$P_{\bar{\sigma}}(\hat{x}^k, \hat{y}) - P_{\bar{\sigma}}((\hat{x}^k, \hat{y}) + d_c^k) \;=\; \hat{\Phi}_c^k(0) - \hat{\Phi}_c^k((d_c^k)_x) + o(\Delta_c^k). \tag{3.117}$$

(3.115) and (3.117) imply $r_c^k = 1 + \dfrac{o(\Delta_c^k)}{\hat{\delta}\Delta_c^k} \to 1$. This shows that $\Delta_c^{k+1} \ge \Delta_c^k$ for all k sufficiently large, since the Δ_c^k is not decreased for $r_c^k \ge 0.1$. This contradicts

$\sum_{k=1}^{\infty} \Delta_c^k \leq \infty$. As a consequence, the iteration sequence $(\hat{x}^k, \hat{y})$ is not bounded away from a stationary point of NLP($\hat{y}$) specified in Definition 2.12, if $\hat{y} \in \hat{Y}$ holds and therefore, a ε-stationary point according to Definition 3.1 of NLP($\hat{y}$) given by (3.6) is obtained after a finite number of iterations for $\hat{y} \in \hat{Y}$ for an arbitrary small $\varepsilon > 0$. This implies, that conditions (3.61) and (3.62) are satisfied after a finite number of iterations. □

Subsuming all previous results, yields the final convergence theorem.

Theorem 3.1. *If Assumption 3.1 and Assumption 3.2 hold, then Algorithm 3.1 terminates after a finite number of iterations at an ε-solution of the convex MINLP* (3.1) *according to Definition 3.3 or it detects that no feasible point exists.*

Proof. Together with Corollary 3.7, Lemma 3.3 and Lemma 3.4 prove Theorem 3.1. □

3.3 Aspects of Implementation and Future Research

The implementation aspects, that are to be considered in this section deal with the update of the matrix B^k in Step 5 of Algorithm 3.1. Furthermore, SOC-steps can be included in Algorithm 3.1 to ensure superlinear convergence for solving NLP($y^{\hat{k}}$) given by (3.6) for some $y^{\hat{k}} \in Y$, if no mixed-integer search steps are carried out.

Algorithm 3.1 requires a symmetric, positive definite $n \times n$ matrix B^k in each iteration k. A suitable possibility to define the matrix is to apply a quasi-Newton update formula in each iteration. Typical update formulas, such as formula (2.21), require Lagrangian multipliers, whenever an update of the matrix is performed. In case of a continuous search step, i.e., if d_i^k determined in Step 3 is not an improving mixed-integer search direction according to Definition 3.2, the multipliers can be obtained directly from the solution of QP(x^k, y^k) given by (3.12) providing the continuous search direction d_c^k in Step 2.

However, we also need to determine Lagrangian multipliers, if a mixed-integer search step is performed, i.e., if d_i^k determined in Step 3 is an improving mixed-integer search direction according to Definition 3.2. Without additional computational effort, multipliers could be taken from the continuous branch-and-bound subproblem, that provided the optimal solution of MIQP(x^k, y^k) given by (3.50). The drawback in this case is, that the values of the multipliers depend on the execution mode of the MIQP solver, that is applied to solve problem (3.50). This means, that the dual part of the solution depends, e.g., on the node-selection strategy or on cut generators, see Section 2.11 and Chapter 4. To avoid such dependency, multipliers can be determined by solving the subsequent constrained least squares problem, which is derived from

the KKT conditions of the continuous relaxation of problem (3.50), where the trust region constraint and the non-negativity constraint for η are neglected.

$$\begin{aligned} &\lambda \in \mathbb{R}^m : \\ &\begin{array}{lrcl} \min & \left\| \nabla_{x,y} f(x^k + (d_i^k)_x, y^k + (d_i^k)_y) - [\nabla_{x,y} g(x^k + (d_i^k)_x, y^k + (d_i^k)_y)]^T \lambda \right\|_2^2 & & \\ \text{s.t.} & \lambda_j & \geq & 0, \quad \forall j \in \mathbb{A}_i^k, \\ & \lambda_j & = & 0, \quad \forall j \in \mathbb{J} \backslash \mathbb{A}_i^k, \end{array} \end{aligned} \tag{3.118}$$

where $\mathbb{A}_i^k \subset \mathbb{J}$ denotes the set of active constraints at the solution d_i^k of problem (3.50) in iteration k. $[\nabla_{x,y} g(x^k + (d_i^k)_x, y^k + (d_i^k)_y)] \in \mathbb{R}^m \times \mathbb{R}^n$ denotes the Jacobian matrix of the constraints.

To ensure superlinear local convergence properties for solving NLP(y^k) given by (3.6) for some fixed $y^k \in Y$, well-known SOC-steps can be incorporated in Algorithm 3.1 as proposed by Yuan [112], if the quality of the model is not sufficient, e.g., $r_c^k < 0.75$ in Step 4. In this case it is necessary to solve an additional subproblem given by

$$\begin{aligned} &(\hat{d}_c)_x \in \mathbb{R}^{n_c} : \\ &\begin{array}{lrcl} \min & \bar{\Phi}^k((\hat{d}_c)_x) & & \\ \text{s.t.} & \|(\bar{d}_c)_x\|_\infty & \leq & \Delta_c^k, \end{array} \end{aligned} \tag{3.119}$$

with

$$\begin{aligned} \bar{\Phi}^k((\hat{d}_c)_x) \;:=\; & \nabla_x f(x^k, y^k)^T (\bar{d}_c)_x + \frac{1}{2} (\bar{d}_c)_x^T B_c^k (\bar{d}_c)_x \\ + \; & \sigma^k \| (g((x^k, y^k) + (d_c^k)_x) + [\nabla_{x,y} g(x^k, y^k)]^T (\hat{d}_c)_x)^- \|_\infty, \end{aligned} \tag{3.120}$$

$$(\bar{d}_c)_x \;:=\; (d_c^k)_x + (\hat{d}_c)_x \tag{3.121}$$

and

$$\hat{d}_c \;:=\; \begin{pmatrix} (\hat{d}_c)_x \\ 0 \end{pmatrix}, \tag{3.122}$$

where $(d_c^k)_x$ is part of the optimal solution $((d_c^k)_x, \eta_c^k)$ of QP(x^k, y^k) given by (3.12) and $(\hat{d}_c)_x \in \mathbb{R}^{n_c}$.

Future research aims at developing an algorithm, that on the one hand solely solves MIQP subproblems, while on the other hand convergence properties at least for convex MINLPs can be established. In the remainder of this section, we want to point out, how such an algorithm could be designed.

In Section 2.10 we reviewed an extension of the trust region method of Yuan, see Algorithm 2.1 for mixed-integer optimization problems, where the continuous problems (2.31) are replaced by mixed-integer quadratic programs, such as (3.50). The resulting implementation is called MISQP, see Exler and Schittkowski [45]. On a large

test set of MINLP problems, it obtains solutions of high quality, i.e., feasible solutions with an objective value close to the best-known one, even for non-convex problems. Moreover, it is very efficient in terms of the number of function evaluations, see Chapter 6. Unfortunately, it was not possible to prove convergence properties for convex MINLP problems, yet.

In Section 3.1 we presented an extension of linear outer approximation, that applies mixed-integer search steps, motivated by MISQP. Convergence of the proposed method is ensured by incorporating linear outer approximations.

Of course, we would like to prove convergence for a MISQP-type method directly without relying on linear outer approximation. Now we want to present a concept to design a convergent method based on the successive solution of MIQP problems (3.50).

Recalling early branching according to Leyffer [76], see Section 2.9, we can derive a stopping criterion for such a method. We consider the convex MINLP (3.1). If we apply early branching, then an extended MINLP given by

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{N}^{n_i}: \\ & \qquad \begin{aligned} \min \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \geq 0 \\ & f(x,y) \leq \hat{\eta} - \varepsilon. \end{aligned} \end{aligned} \tag{3.123}$$

is solved. $\hat{\eta} \in \mathbb{R} \cup \{\infty\}$ denotes the incumbent, i.e., the feasible solution possessing the lowest objective function value among all feasible solutions found so far. A quadratic approximation of MINLP (3.123) is given by the subsequent MIQP problem, where bounds on x and y are implicitly included:

$$\begin{aligned} & d_x \in \mathbb{R}^{n_c}, \quad d_y \in \mathbb{N}^{n_i}: \\ & \qquad \begin{aligned} \min \quad & \nabla_{x,y} f(x^k,y^k)^T \begin{pmatrix} d_x \\ d_y \end{pmatrix} + \frac{1}{2}(d_x^T, d_y^T) B^k \begin{pmatrix} d_x \\ d_y \end{pmatrix} \\ \text{s.t.} \quad & g(x^k,y^k) + \left[\nabla_{x,y} g(x^k,y^k)\right]^T \begin{pmatrix} d_x \\ d_y \end{pmatrix} \geq 0 \\ & f(x^k,y^k) + \nabla_{x,y} f(x^k,y^k)^T \begin{pmatrix} d_x \\ d_y \end{pmatrix} \leq \hat{\eta} - \varepsilon \end{aligned} \end{aligned} \tag{3.124}$$

Corollary 3.8. *Let Assumption 3.1 hold. Then $\hat{\eta}$ is the optimal solution of the convex MINLP* (3.1) *with respect to termination tolerance $\varepsilon > 0$, if MIQP* (3.124) *is infeasible.*

Proof. Applying

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x^k \\ y^k \end{pmatrix}, \tag{3.125}$$

proves the corollary, since the feasible region of MIQP (3.124) is a relaxation the feasible region of MINLP (3.123), derived from a convex MINLP (3.1). □

The subsequent corollary is a first step towards a convergent sequential mixed-integer quadratic algorithm. Convergence could rely on branch-and-bound enumeration on MIQP subproblem level.

Corollary 3.9. *Let Assumption 3.1 hold. Then any integer value included in an infeasible subtree of a branch-and-bound enumeration of MIQP* (3.124) *is non-optimal for the convex MINLP* (3.1)*.*

Proof. *Applying Lemma 2.7, see Section 2.9.*

Further research is needed to obtain a complete algorithm based on the previous corollaries, for which convergence properties for convex MINLPs can be established.

4. REVIEW ON DISJUNCTIVE CUTTING PLANES FOR MILPS

Cutting planes, which were introduced in a famous paper of Gomory [59] in 1958, possess two properties. On the one hand they cut off the solution $(\bar{x}, \bar{y})$ of a relaxation of the original problem, which leads to a truncation of the feasible region of the corresponding relaxation. On the other hand, cutting planes do not alter the solution of the original problem. If an inequality possesses these properties, it is called a valid cutting plane. In the context of mixed-integer linear and quadratic programming a cutting plane is linear. As a cutting plane cuts off the solution of a relaxation of the original problem, this relaxation leads to an improved lower bound, if the cutting plane is added to the set of constraints of the relaxation. Considering general mixed-integer nonlinear programming, cutting planes can be used to solve convex MINLP problems, see Section 2.7.

The combination of a branch-and-bound enumeration procedure, see Section 2.4, and cutting plane generation is called branch-and-cut. Branch-and-cut is the state-of-the-art solution method for mixed-integer linear optimization problems. Cutting planes lead to a tremendous speed-up of MILP solvers according to Bixby [26], see Table 1.1. Table 1.1 evaluates the additional techniques, that improved the performance of a pure branch-and-bound method. It shows that the generation of cutting planes lead to a speed-up factor of over 50.

Although cutting planes have been applied successfully within branch-and-cut solvers for mixed-integer linear programming, there are almost no results on the generation of cutting planes for mixed-integer quadratic programming. As a consequence, the application of cutting planes within a MIQP branch-and-cut solver is analyzed in Chapter 5. In this chapter we introduce well-known theory on cutting planes for mixed-integer linear programs focusing on disjunctive cutting planes.

In general, there are two different kinds of cutting planes. The first class consists of general cutting planes, while the second one exploits specific problem structure yielding so-called structural cutting planes. Structural cutting planes can be constructed for MIQPs without further adjustments, if the corresponding problem structure is identified. In contrast, general cutting planes do not rely on any structure of the feasible region.

In this thesis we focus on MIQP problems of form

$$
\begin{aligned}
& x \in X,\ y \in Y: \\
& \qquad \min \quad \frac{1}{2}(x^T, y^T)\, B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix} \\
& \qquad \text{s.t.} \quad A_E \begin{pmatrix} x \\ y \end{pmatrix} = b_E, \\
& \qquad\qquad\ \ A_I \begin{pmatrix} x \\ y \end{pmatrix} \geq b_I.
\end{aligned}
\tag{4.1}
$$

x and y denote the vectors of the continuous and integer variables, respectively, while $B \in \mathbb{R}^{n \times n}$ is a positive definite matrix and $c \in \mathbb{R}^n$ holds. X and Y are defined by the upper and lower bounds on both the continuous and the integer variables, see (1.2). n_c denotes the number of continuous variables and n_i is the number of integer variables. The total number of variables is denoted by n, i.e., $n := n_i + n_c$. Equality constraints are denoted by $A_E \in \mathbb{R}^{m_e \times n}$ and $b_E \in \mathbb{R}^{m_e}$, while inequality constraints are given by $A_I \in \mathbb{R}^{m_i \times n}$ and $b_I \in \mathbb{R}^{m_i}$. Therefore m_e denotes the number of equality constraints, while m_i is the number of inequality constraints.

MIQP (4.1) arises as subproblem of general unstructured MINLP problems generated by MISQP, see Section 2.10, or MIQPSOA, see Chapter 3. In this case the feasible region of MIQP (4.1) is determined by linearizations of arbitrary nonlinear constraints, and therefore we cannot assume, that the constraints of the MIQP possess any specific structure. As a consequence, we focus on general cutting planes.

Most techniques for constructing general cutting planes rely on the property, that the solution of the relaxation to be cut off is a basic solution, i.e., a corner of the feasible polyhedral set described by the constraints of problem (4.1). In general, the solutions of most continuous quadratic programs, obtained by a branch-and-bound enumeration during the solution of MIQP (4.1) are non-basic solutions. This may be the main reason, why there exist only very few results on applying cutting planes for solving mixed-integer quadratic programming problems of form (4.1) by QP-based branch-and-bound. The only result we are aware of, is Bienstock [25], who suggests a branch-and-cut method based on disjunctive cutting planes for special MIQP problems, arising in portfolio optimization. Although disjunctive cutting planes belong to the class of general cutting planes, [25] focuses on structural cutting planes, by exploiting the structure of one single constraint for cut generation. See also Section 4.2 for a general introduction on disjunctive cutting planes.

Currently some research deals with the construction of disjunctive cutting planes for the convex MINLP problem (1.6), see e.g., Grossmann and Lee [63] or Kilinc, Linderoth and Luedtke [68]. The corresponding results differ from the research presented in this thesis. Present research reformulates (4.1), such that the objective function becomes linear, while here the property, that the feasible region is a polyhedron, is exploited to efficiently construct cutting planes, see Chapter 5.

In mixed-integer linear programming, the construction of the convex hull of all fea-

sible integer points is of special interest. If it is known, the solution of the mixed-integer linear program is equivalent to the solution of its continuous relaxation, see e.g. Krumke [69]. This is not the case for mixed-integer quadratic programs of form (4.1). In general, the solution of the continuous relaxation is non-integral, even if the feasible region is the convex hull of all integral points. The difference is, that in linear programming one of the optimal solutions is always basic, i.e., a basic solution. If the feasible region is given by the convex hull of all integer feasible points, every optimal basic solution of the linear relaxation is integral and therefore optimal for the corresponding mixed-integer linear program. In contrast, the optimal solution of the continuous relaxation of MIQP (4.1), is the feasible point with minimal distance to the minimizer of the quadratic objective function measured in the $\|.\|_{\mathsf{B}}$-Norm. This can be any point on the boundary or even inside the convex hull of the integer feasible points. As a consequence, it needs not be integral. Note, that the norm $\|.\|_{\mathsf{B}}$ is determined by the matrix B of the objective function of MIQP (4.1).

Nevertheless, cutting planes can be beneficial for mixed-integer quadratic programming, since they lead to a tighter relaxation. This gives rise to better lower bounds during the branch-and-bound enumeration process, and therefore they may lead to a smaller branch-and-bound tree. Since the convex hull of the integer feasible points cannot be constructed in polynomial time, see e.g. Cornuejols [40], the same is true for general MILPs from a practical point of view.

If cutting planes are only generated at the root node of the branch-and-bound search tree, they are valid for each subproblem within the enumeration process. In principle, the cut generation routines can be applied at any node of the branch-and-bound tree, but an efficient cut management and cut selection procedure is required, since the constructed cutting planes are only locally valid.

At the root node of the branch-and-bound tree cutting planes are usually generated iteratively in rounds. First, a relaxation of the original problem is solved and then cutting planes are constructed. Afterwards, they are added to the set of constraints and the tightened relaxation is solved again. One cycle consisting of the solution of the relaxation and the generation of cutting planes is called round. This process is continued until no more cuts are found or the maximal number of rounds is reached.

In the remainder of the chapter we focus on disjunctive cutting planes. First we review briefly some basic definitions from linear programming. Then disjunctive programming and disjunctive cutting planes are introduced, which were both founded by Balas [10]. Furthermore, we review simple disjunctive cutting planes and their connection to disjunctive cutting planes, which was established by Balas and Perregaard [17]. Finally we present an efficient cut generation procedure for disjunctive cutting planes, which goes back to Perregaard [86] and which is to be extended for MIQPs in Chapter 5.

4.1 Linear Programming Basics

In this section, we provide some basic definitions of linear programming, that are needed to describe and derive the theory on disjunctive cutting planes, see e.g., Jarre and Stoer [66] for more details. We consider the continuous linear program obtained from the continuous relaxation of MIQP (4.1) by neglecting the quadratic term in the objective function

$$\begin{aligned} & x \in X,\ y \in Y_{\mathbb{R}} : \\ & \begin{array}{rll} \min & c^T \begin{pmatrix} x \\ y \end{pmatrix} & \\ \text{s.t.} & A_E \begin{pmatrix} x \\ y \end{pmatrix} & = \ b_E, \\ & A_I \begin{pmatrix} x \\ y \end{pmatrix} & \geq \ b_I. \end{array} \end{aligned} \tag{4.2}$$

I.e., $c \in \mathbb{R}^n$, $A_E \in \mathbb{R}^{m_e \times n}$, $b_E \in \mathbb{R}^{m_e}$, $A_I \in \mathbb{R}^{m_i \times n}$, and $b_I \in \mathbb{R}^{m_i}$, where $n := n_c + n_i$ denotes the total number of variables and m_e denotes the number of equality constraints, while m_i is the number of inequality constraints.

LP (4.2) can be transformed to the so-called standard form given by

$$\begin{aligned} & x \in \mathbb{R}^{\tilde{n}_c}, y \in \mathbb{R}^{n_i} : \\ & \begin{array}{rll} \min & \tilde{c}^T \begin{pmatrix} \tilde{x} \\ y \end{pmatrix} & \\ \text{s.t.} & \tilde{A} \begin{pmatrix} \tilde{x} \\ y \end{pmatrix} & = \ \tilde{b} \\ & \begin{pmatrix} \tilde{x} \\ y \end{pmatrix} & \geq \ 0, \end{array} \end{aligned} \tag{4.3}$$

with $\tilde{A} \in \mathbb{R}^{\tilde{m} \times \tilde{n}}$, $\tilde{c} \in \mathbb{R}^{\tilde{n}}$ and $\tilde{b} \in \mathbb{R}^{\tilde{m}}$, where $\tilde{n}$ denotes the total number of variables and $\tilde{m}$ denotes the number of equality constraints. The index set of the continuous variables $\tilde{x}$ is denoted by $J \subset \{1, \ldots, \tilde{n}\}$. The index set of the integer variables y within $\{1, \ldots, \tilde{n}\}$ is represented by I. LP (4.3) is obtained from LP (4.2) by a variable transformation, such that $x_l = 0$ and $y_l = 0$ holds. Furthermore, one slack variable is introduced for each inequality constraint, where the slack variables are part of the continuous variables $\tilde{x}$. Upper box-constraints are contained in the constraints of LP (4.3) yielding $\tilde{m} > m_E + m_I$ equality constraints after the introduction of further slack variables. Unrestricted variables are split into a positive and a negative variable. Due to these additional variables and due to the introduction of slack variables, the total number of variables $\tilde{n}$ is larger than the number of variables n of LP (4.2).

The following definition of a basis and a basic solution is very important in linear programming, e.g., in the context of general cutting planes but also for the simplex method, one of the main solution techniques for linear programs.

Definition 4.1. *Let* $\tilde{A} \in \mathbb{R}^{\tilde{m} \times \tilde{n}}$ *be the matrix corresponding to a system of* $\tilde{m}$ *linear equations in* $\tilde{n}$ *variables. The index set* B *of columns of* $\tilde{A}$ *is a basis of* $\tilde{A}$*, if* $|B| = \tilde{m}$ *and if the corresponding sub-matrix* $\tilde{A}_B := [\tilde{a}^j]_{j \in B}$ *is invertible.* $\tilde{a}^j$ *denotes the* j*-th column of* $\tilde{A}$*. The complementary index set* $B^\perp$ *with* $B \cap B^\perp = \emptyset$ *and* $B \cup B^\perp = I \cup J$ *contains the so-called non-basic variables.*

If the system of linear equations

$$\tilde{A} \begin{pmatrix} \tilde{x} \\ y \end{pmatrix} = \tilde{b} \tag{4.4}$$

is feasible, then $\tilde{A}$ possesses at least one basis B and the corresponding non-basic index-set $B^\perp$. Furthermore, the solution of system (4.4), which is often denoted by $\bar{a}_0$ can be expressed in terms of B and $B^\perp$, since

$$\bar{a}_0 = \begin{pmatrix} \tilde{x}_B \\ y_B \end{pmatrix} + \bar{A}_{B^\perp} \begin{pmatrix} \tilde{x}_{B^\perp} \\ y_{B^\perp} \end{pmatrix} \tag{4.5}$$

holds due to

$$\tilde{b} = \tilde{A} \begin{pmatrix} \tilde{x} \\ y \end{pmatrix} = \tilde{A}_B \begin{pmatrix} \tilde{x}_B \\ y_B \end{pmatrix} + \tilde{A}_{B^\perp} \begin{pmatrix} \tilde{x}_{B^\perp} \\ y_{B^\perp} \end{pmatrix} \tag{4.6}$$

with

$$\bar{a}_0 := \tilde{A}_B^{-1} \tilde{b} \tag{4.7}$$

$$\bar{A}_{B^\perp} := \tilde{A}_B^{-1} \tilde{A}_{B^\perp}. \tag{4.8}$$

Definition 4.2. *The solution of system* (4.4) *with*

$$\begin{pmatrix} \tilde{x}_{B^\perp} \\ y_{B^\perp} \end{pmatrix} = 0 \tag{4.9}$$

and

$$\begin{pmatrix} \tilde{x}_B \\ y_B \end{pmatrix} = \bar{a}_0 \tag{4.10}$$

is called basic solution of the basis B*. If system* (4.4) *belongs to LP* (4.3) *and*

$$\begin{pmatrix} \tilde{x}_B \\ y_B \end{pmatrix} \geq 0 \tag{4.11}$$

holds, then B *is called feasible basis of LP* (4.3) *and* $\left(\begin{pmatrix} \tilde{x}_B \\ y_B \end{pmatrix}, \begin{pmatrix} \tilde{x}_{B^\perp} \\ y_{B^\perp} \end{pmatrix} \right)$ *is called a feasible, basic solution.*

For reviewing and extending the theory on disjunctive programming, we use a slightly different formulation for a linear program, which also leads to a different but equivalent definition of a basis. We introduce subsequent notation, since on the one hand it is consistent with the notation used by Balas, Perregaard and others working on disjunctive programming. On the other hand, it is more compatible with the notation used in mixed-integer nonlinear programming, where, e.g., bounds on the variables play an important role.

We consider the following linear program

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{R}^{n_i}: \\ & \qquad \min \quad c^T \begin{pmatrix} x \\ y \end{pmatrix} \\ & \qquad \text{s.t.} \quad A \begin{pmatrix} x \\ y \end{pmatrix} \geq b, \\ & \qquad\qquad \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \end{aligned} \tag{4.12}$$

with $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, where $n := n_c + n_i$ denotes the total number of variables and m denotes the number of constraints. Note, that LP (4.12) is equivalent to LP (4.2) after a variable transformation yielding

$$\begin{pmatrix} x_l \\ y_l \end{pmatrix} = 0 \tag{4.13}$$

and after including the upper bounds on the variables in the constraints where equality constraints are formulated as inequality constraints, i.e.,

$$A := \begin{bmatrix} A_I \\ A_E \\ -A_E \\ -I \end{bmatrix}, \qquad b := \begin{bmatrix} b_I \\ b_E \\ -b_E \\ -\begin{pmatrix} x_u \\ y_u \end{pmatrix} \end{bmatrix}. \tag{4.14}$$

After also including the lower bounds within the constraints, LP (4.12) has the subsequent form

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{R}^{n_i}: \\ & \qquad \min \quad c^T \begin{pmatrix} x \\ y \end{pmatrix} \\ & \qquad \text{s.t.} \quad \hat{A} \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}, \end{aligned} \tag{4.15}$$

where $\hat{A} \in \mathbb{R}^{\hat{m} \times n}$ and $\hat{b} \in \mathbb{R}^{\hat{m}}$ are therefore given by

$$\hat{A} := \begin{bmatrix} A \\ I \end{bmatrix}, \tag{4.16}$$

$$\hat{b} := \begin{bmatrix} b \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{bmatrix}, \tag{4.17}$$

with $\hat{m} := m + n$. In this formulation $\hat{m} \geq n$ holds, due to the existence of upper and lower bounds for every variable. LP formulation (4.15) motivates a different definition of a basis and a basic solution.

Definition 4.3. *Let $\hat{A} \in \mathbb{R}^{\hat{m} \times n}$ be the matrix corresponding to a system of $\hat{m}$ linear inequalities in n variables with $\hat{m} \geq n$. The index set B of rows of $\hat{A}$ is a basis of $\hat{A}$, if $|B| = n$ and if the corresponding sub-matrix $\hat{A}_B := [\hat{a}_j]_{j \in B}$ is invertible, where $\hat{a}_j$ denotes the j-th row of $\hat{A}$.*

A solution $(\bar{x}, \bar{y})$ of the system

$$\hat{A} \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b} \tag{4.18}$$

is called a basic solution, if n linear independent constraints are satisfied as equalities at $(\bar{x}, \bar{y})$, i.e., the corresponding rows of $\hat{A}$ form a basis. If all constraints of the system (4.18) are satisfied by the basic solution $(\bar{x}, \bar{y})$, $(\bar{x}, \bar{y})$ is called a feasible, basic solution.

The subsequent corollary shows the relation of Definition 4.1 and 4.3.

Corollary 4.1. *Let $(\bar{x}, \bar{y})$ be a basic solution according to Definition 4.3. Then $(\bar{x}, \bar{y})$ can be extended yielding a basic solution according to Definition 4.2.*

Proof. Let $(\bar{x}, \bar{y})$ be a basic solution according to Definition 4.3. Then n linear independent constraints of the linear system (4.18) are active at $(\bar{x}, \bar{y})$. We denote the corresponding submatrix by $\hat{A}_B$, where B is the index set of the active constraints.

Now we transform the linear system (4.18) into standard form by introducing slack variables $s \in \mathbb{R}^{\hat{m}}$. The system becomes

$$\hat{A} \begin{pmatrix} x \\ y \end{pmatrix} - s = \hat{b}, \tag{4.19}$$

$$\begin{pmatrix} x \\ y \\ s \end{pmatrix} \geq 0, \tag{4.20}$$

doubling the constraints

$$\begin{pmatrix} x \\ y \end{pmatrix} \geq 0. \tag{4.21}$$

Each slack variable can be classified according to whether or not the corresponding constraint is active at $(\bar{x}, \bar{y})$. Since the active constraints are indexed by B, we denote the corresponding slack variables by s_B. The remaining slack variables are denoted by $s_{B^\perp}$. Considering the basic constraints, which are active at $(\bar{x}, \bar{y})$, we define $\bar{s}_B := 0$, while we set

$$\bar{s}_{B^\perp} := \hat{A}_{B^\perp} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_{B^\perp}. \tag{4.22}$$

for non-basic constraints.

Note, that some constraints from the set $B^\perp$ might also be active in addition to the basic constraints indexed by B.

The linear system (4.19) possesses $\hat{m}$ equations in total and contains n original variables (x, y) and $\hat{m}$ slack variables s, with $n < \hat{m}$. The solution $(\bar{x}, \bar{y}, \bar{s}_{B^\perp}, \bar{s}_B)$ of (4.19) is a basic solution for the basis $\tilde{B}$, with basic variables $(\bar{x}, \bar{y}, \bar{s}_{B^\perp})$ and non-basic variables s_B satisfying $\bar{s}_B = 0$. The matrix representing the linear system (4.19) for all variables $(x, y, s_{B^\perp}, s_B)$ is given by

$$\tilde{A} := \begin{bmatrix} \hat{A} & -I \end{bmatrix} = \begin{bmatrix} \hat{A}_B & 0 & -I \\ \hat{A}_{B^\perp} & -I & 0 \end{bmatrix}. \tag{4.23}$$

Therefore the basis matrix for the basis $\tilde{B}$ according to Definition 4.1 is given by

$$\tilde{A}_{\tilde{B}} = \begin{bmatrix} \hat{A}_B & 0 \\ \hat{A}_{B^\perp} & -I \end{bmatrix}, \tag{4.24}$$

which is invertible, since $\hat{A}_B$ is invertible. □

For simplification of subsequent calculations in this chapter as well as in Chapter 5, we introduce the standard notation

$$\begin{aligned} \bar{a}_{kj} &:= -(e_k^T \hat{A}_B^{-1})_j, \\ \bar{a}_{ij} &:= -(\hat{a}_i^T \hat{A}_B^{-1})_j, \\ \bar{a}_{k0} &:= (e_k^T \hat{A}_B^{-1} \hat{b}_B), \\ \bar{a}_{i0} &:= (\hat{a}_i^T \hat{A}_B^{-1} \hat{b}_B - \hat{b}_i). \end{aligned} \tag{4.25}$$

4.2 Introduction on Disjunctive Cutting Planes

In this section we introduce some theory for disjunctive cutting planes and the closely related disjunctive programs. It was developed since the early seventies by Balas [10] and others. Latest results by Perregaard [86] and Balas and Bonami [18], turn these cutting planes into very powerful cut generators for mixed-integer linear solvers. Our

aim is to extend the results, such that disjunctive cutting planes can be applied efficiently for mixed-integer quadratic programs, see Chapter 5.

Reviewing the results of Perregaard [86], we consider a mixed-integer linear program

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{N}^{n_i}: \\ & \begin{aligned} \min \quad & c^T \begin{pmatrix} x \\ y \end{pmatrix} \\ \text{s.t.} \quad & A \begin{pmatrix} x \\ y \end{pmatrix} \geq b, \\ & \begin{pmatrix} x \\ y \end{pmatrix} \geq 0, \end{aligned} \end{aligned} \tag{4.26}$$

with $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, where $n := n_c + n_i$ denotes the total number of variables and m denotes the number of constraints. Note, that the constraints contain lower and upper bounds for each variable, see also Section 4.1. The optimal solution of (4.26) is given by $(x^*, y^*) \in \mathbb{R}^{n_c} \times \mathbb{N}^{n_i}$. The corresponding continuous relaxation of MILP (4.26) is given by LP (4.12).

Let $(\bar{x}, \bar{y}) \in \mathbb{R}^{n_c+n_i}$ be the solution of the continuous relaxation (4.12). Moreover, let the k-th component of $\bar{y}$ be fractional, i.e., $\bar{y}_k \notin \mathbb{N}$. As a consequence, either $y_k^* \geq \lceil \bar{y}_k \rceil$ or $y_k^* \leq \lfloor \bar{y}_k \rfloor$ holds, since $(x^*, y^*) \in \mathbb{R}^{n_c} \times \mathbb{N}^{n_i}$ is the optimal solution of problem (4.26) with y_k^* integral. This property can be expressed via a logical condition, which is called a disjunction $\vee$, i.e.,

$$y_k^* \geq \lceil \bar{y}_k \rceil \quad \vee \quad y_k^* \leq \lfloor \bar{y}_k \rfloor. \tag{4.27}$$

The feasible region of the continuous relaxation (4.12) is truncated by integrating the disjunctive condition (4.27). The resulting relaxation of MILP (4.26) is called disjunctive relaxation and is given by

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{R}^{n_i}: \\ & \begin{aligned} \min \quad & c^T \begin{pmatrix} x \\ y \end{pmatrix} \\ \text{s.t.} \quad & A \begin{pmatrix} x \\ y \end{pmatrix} \geq b \\ & \begin{pmatrix} x \\ y \end{pmatrix} \geq 0 \\ & y_k \geq \lceil \bar{y}_k \rceil \vee y_k \leq \lfloor \bar{y}_k \rfloor \ . \end{aligned} \end{aligned} \tag{4.28}$$

The additional constraint

$$y_k \geq \lceil \bar{y}_k \rceil \quad \vee \quad y_k \leq \lfloor \bar{y}_k \rfloor \tag{4.29}$$

is called disjunctive condition. The solution of the disjunctive relaxation (4.28) is denoted by $(\bar{x}_{DJ}, \bar{y}_{DJ}) \in \mathbb{R}^{n_c+n_i}$. Since $(\bar{x}, \bar{y})$ is not feasible for the disjunctive relaxation (4.28) and LP (4.12) is a relaxation of (4.28),

$$c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \leq c^T \begin{pmatrix} \bar{x}_{DJ} \\ \bar{y}_{DJ} \end{pmatrix} \leq c^T \begin{pmatrix} x^* \\ y^* \end{pmatrix} \tag{4.30}$$

holds. The feasible region of the disjunctive relaxation (4.28) is the union of the two polyhedra

$$P^k_{\text{lower}} := \left\{ A \begin{pmatrix} x \\ y \end{pmatrix} \geq b,\ \begin{pmatrix} x \\ y \end{pmatrix} \geq 0,\ y_k \leq \lfloor \bar{y}_k \rfloor \right\} \tag{4.31}$$

and

$$P^k_{\text{upper}} := \left\{ A \begin{pmatrix} x \\ y \end{pmatrix} \geq b,\ \begin{pmatrix} x \\ y \end{pmatrix} \geq 0,\ y_k \geq \lceil \bar{y}_k \rceil \right\}. \tag{4.32}$$

An optimization problem possessing a linear objective and a feasible region given by the union of finitely many polyhedra, is called a disjunctive program. We denote the finite index set of the corresponding polyhedra by Q, i.e., $P^h, h \in Q$ is the corresponding h-th polyhedron. For the disjunctive relaxation (4.28) on variable $y_k \in \mathbb{N}$, $|Q| = 2$ and $Q = \{1, 2\}$ with $P^1 = P^k_{\text{lower}}$ and $P^2 = P^k_{\text{upper}}$ holds.

The disjunctive relaxation (4.28) can be used to generate cutting planes for the mixed-integer problem (4.26). For the construction of cutting planes, the existence of a compact representation of the convex hull of the union of polyhedra in a higher dimension is exploited. The associated convex hull can be projected on the original space yielding so-called disjunctive cutting planes. We introduce some related definitions and theorems.

Definition 4.4. *An inequality*

$$a_c^T \begin{pmatrix} x \\ y \end{pmatrix} \geq b_c, \tag{4.33}$$

with $a_c \in \mathbb{R}^n,\ b_c \in \mathbb{R}$ *is said to be a consequence of the linear system*

$$A \begin{pmatrix} x \\ y \end{pmatrix} \geq b, \tag{4.34}$$

if (4.33) *is satisfied by every* $(x, y) \in \mathbb{R}^{n_c} \times \mathbb{R}^{n_i}$*, which satisfies the linear system* (4.34)*.*

The following theorem of Balas [11] explains the correlation between disjunctive cutting planes and the disjunctive relaxation (4.28).

Theorem 4.1. *Let* $Q^* \neq \emptyset$ *be a index set of non-empty polyhedra* $P^h,\ h \in Q$*, where* Q *is the index set of all polyhedra, i.e., for* $P^h,\ h \in Q^*$

$$P^h := \left\{ (x, y) \in \mathbb{R}^n :\ A \begin{pmatrix} x \\ y \end{pmatrix} \geq b,\ \begin{pmatrix} x \\ y \end{pmatrix} \geq 0,\ D^h \begin{pmatrix} x \\ y \end{pmatrix} \geq d_0^h \right\} \neq \emptyset \tag{4.35}$$

holds, with $D^h \in \mathbb{R}^{m_h \times n}$ *and* $d_0^h \in \mathbb{R}^{m_h}$*.* $m^h \in \mathbb{N}$ *with* $m^h > 0$ *is the dimension of the disjunctive conditions*

$$D^h \begin{pmatrix} x \\ y \end{pmatrix} \geq d_0^h. \tag{4.36}$$

The inequality $a_c^T \begin{pmatrix} x \\ y \end{pmatrix} \geq b_c$ *with* $a_c \in \mathbb{R}^n$ *and* $b_c \in \mathbb{R}$ *is a consequence of the constraints*

$$\begin{aligned} A \begin{pmatrix} x \\ y \end{pmatrix} &\geq b \\ \begin{pmatrix} x \\ y \end{pmatrix} &\geq 0 \\ \vee_{h \in Q} \left(D^h \begin{pmatrix} x \\ y \end{pmatrix} &\geq d_0^h \right), \end{aligned} \tag{4.37}$$

if and only if there exists a set of vectors $u^h \in \mathbb{R}^m,\ u_0^h \in \mathbb{R}^{m^h},\ u^h,\ u_0^h \geq 0,\ h \in Q^*$, *such that*

$$\begin{aligned} a_c &\geq A^T u^h + (D^h)^T u_0^h, \quad \forall h \in Q^*, \\ b_c &\leq b^T u^h + (d_0^h)^T u_0^h, \quad \forall h \in Q^*, \end{aligned} \tag{4.38}$$

holds.

Proof. The proof is closely related to Farkas Lemma, see Balas [11]. □

The disjunctive relaxation (4.28) is called two-term disjunction, since $|Q| = 2$ holds. It is obtained from the general disjunctive program (4.37) by setting

$$D^1 = -e_k^T, \tag{4.39}$$
$$D^2 = e_k^T, \tag{4.40}$$
$$d_0^1 = -\lfloor \bar{y}_k \rfloor, \tag{4.41}$$
$$d_0^2 = \lceil \bar{y}_k \rceil, \tag{4.42}$$

where $e_k \in \mathbb{R}^n$ denotes the k-th unit vector. Note, that P_{lower}^k is the polyhedron described by inequalities

$$\begin{aligned} A \begin{pmatrix} x \\ y \end{pmatrix} &\geq b, \\ \begin{pmatrix} x \\ y \end{pmatrix} &\geq 0, \\ -y_k = D^1 \begin{pmatrix} x \\ y \end{pmatrix} &\geq d_0^1 = -\lfloor \bar{y}_k \rfloor, \end{aligned} \tag{4.43}$$

while P^k_{upper} is given by

$$\begin{aligned} A\begin{pmatrix} x \\ y \end{pmatrix} &\geq b, \\ \begin{pmatrix} x \\ y \end{pmatrix} &\geq 0, \\ y_k = D^2\begin{pmatrix} x \\ y \end{pmatrix} &\geq d_0^2 = \lceil \bar{y}_k \rceil. \end{aligned} \tag{4.44}$$

Disjunctive cutting planes are derived from the disjunctive relaxation (4.28) or more generally (4.37). Theorem 4.1 shows, that any cutting plane $a_c^T \begin{pmatrix} x \\ y \end{pmatrix} \geq b_c$ induced from a disjunctive relaxation has to satisfy conditions (4.38), in order to be valid for the relaxation (4.37).

Therefore, a cutting plane is given by an inequality, which is violated by the solution $(\bar{x}, \bar{y})$ of the continuous relaxation (4.12) and which satisfies the conditions of Theorem 4.1. As a consequence, a cutting plane can be obtained by solving the so-called cut generating linear program ($CGLP_k$) corresponding to disjunction (4.27), see e.g., Balas [11]. Note, that the subsequent formulation of the $CGLP_k$, also contains the lower bounds on the variables by using matrix $\hat{A}$ and right hand side $\hat{b}$ instead of A and b, see LP formulations (4.12) and (4.15).

$$(a_c, b_c) \in \mathbb{R}^n \times \mathbb{R},\ (u, u_0) \in \mathbb{R}^{\hat{m}} \times \mathbb{R},\ (v, v_0) \in \mathbb{R}^{\hat{m}} \times \mathbb{R}:$$

$$\begin{aligned} \min \quad & a_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - b_c \\ \text{s.t.} \quad & a_c - \hat{A}^T u + u_0 e_k \geq 0 \\ & a_c - \hat{A}^T v - v_0 e_k \geq 0 \\ & -b_c + \hat{b}^T u - u_0 \lfloor \bar{y}_k \rfloor = 0 \\ & -b_c + \hat{b}^T v + v_0 \lceil \bar{y}_k \rceil = 0 \\ & \sum_{i=1}^{\hat{m}} u_i + u_0 + \sum_{i=1}^{\hat{m}} v_i + v_0 = 1 \\ & u,\ u_0,\ v,\ v_0 \geq 0. \end{aligned} \tag{4.45}$$

The constraint

$$\sum_{i=1}^{\hat{m}} u_i + u_0 + \sum_{i=1}^{\hat{m}} v_i + v_0 = 1 \ . \tag{4.46}$$

is called normalization constraint. Note, that (u, u_0) correspond to P^k_{lower}, while (v, v_0) are associated with P^k_{upper}, i.e., (u^1, u^1_0) and (u^2, u^2_0) in Theorem 4.1, respectively.

The linear program (4.45) constructs the cutting plane $a_c^T \begin{pmatrix} x \\ y \end{pmatrix} \geq b_c$, that is most violated at the solution $(\bar{x}, \bar{y})$ of the continuous relaxation (4.12), i.e., it minimizes the objective function of the $CGLP_k$. The constraints of problem (4.45) restrict the choice to those cutting planes that are valid for the union of the polyhedra P^k_{lower} and P^k_{upper} defined by (4.31) and (4.32), respectively. The optimization variables (a_c, b_c) correspond to the coefficients of the resulting cutting plane, whereas the variables (u, u_0, v, v_0) ensure the validity of the constructed cut according to Theorem 4.1.

The cut generating linear program (4.45) obviously depends on the disjunction that is considered. In addition it depends on the point to be cut off and therefore forms the objective function, here $(\bar{x}, \bar{y})$. Furthermore, it depends on the constraints of underlying disjunctive program, here determined by the matrix $\hat{A}$ and the right hand side $\hat{b}$. In case we consider different cut generating linear programs, we use the notation $CGLP_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$ in order to allow a precise distinction.

$CGLP_k$ only considers the integrality condition for variable y_k. Therefore, the generated cutting plane can be strengthened by a procedure of Balas and Jeroslow [14], that takes the integrality conditions for additional integer variables $y_i \in I\backslash\{k\}$ into account. Solving $CGLP_k$ yields one cutting plane. There exists various suggestions how to generate more than just one cutting plane from the disjunctive relaxation corresponding to disjunction (4.27), see Perregaard [86] for details. Furthermore, it is possible to neglect those variables x_i, or y_i respectively, that take the value of one of the corresponding box-constraints in the solution $(\bar{x}, \bar{y})$ of LP (4.12), i.e., $\bar{x}_i = (x_u)_i$ or $\bar{x}_i = (x_l)_i$ or $\bar{y}_i = (y_u)_i$ or $\bar{y}_i = (y_l)_i$ respectively. As a consequence, the disjunctive relaxation (4.28) is restricted to a lower dimensional subspace, such that the dimension of $CGLP_k$ is significantly reduced. The cutting plane determined by the corresponding cut generating linear program can be lifted to be valid in the full space, see Perregaard [86].

Further theory on disjunctive programming and related cutting planes can be found in Balas [9, 10, 11, 12], Balas, Ceria, and Cornuejols [13], Balas and Perregaard [15, 16, 17], Ceria and Pataki [37], Ceria and Soares [38] and others.

4.3 Simple Disjunctive Cuts

Before we return to disjunctive cutting planes, we introduce simple disjunctive cutting planes or intersection cuts, see Balas [9]. Simple disjunctive cutting planes are a special case of disjunctive cutting planes. Furthermore, it turned out, that disjunctive cutting planes can efficiently be constructed based on simple disjunctive cuts. Simple disjunctive cuts can be constructed at any basic solution $(\bar{x}, \bar{y}) \in \mathbb{R}^{n_c+n_i}$ with basis B and the non-basic index set denoted by $B^\perp$, see Definition 4.1 or 4.3. Note, that a basic solution corresponds to a corner of the polyhedron representing the feasible region of the continuous relaxation given by LP (4.15). Furthermore, each fractional integer variable $y_k \in Y_{\mathbb{R}}$ with $\bar{y}_k \notin \mathbb{N}$, yields one simple disjunctive cut. As a conse-

quence, simple disjunctive cuts exist for each optimal, non-integral, basic solution of the continuous relaxation (4.15), since one of the optimal solutions of every LP is a basic solution, see e.g., Jarre and Stoer [66]. Note, that simple disjunctive cuts are equivalent to mixed-integer Gomory cutting planes, see Cornuejols [40]. The simple disjunctive cut is defined in terms of the non-basic variables.

Let the corresponding basic matrix of the active constraints be denoted by $\hat{A}_B \in \mathbb{R}^{n\times n}$ with right hand side $\hat{b}_B \in \mathbb{R}^n$. The non-basic variables are the slack variables for the active constraints forming $\hat{A}_B$ and are denoted by $s_B \in \mathbb{R}^n$. Since $(\bar{x}, \bar{y}) \in \mathbb{R}^{n_c+n_i}$ is by assumption a basic solution, $\bar{s}_B = 0$ holds, see Definition 4.2. Considering the active constraints only, we obtain

$$\begin{aligned} \hat{A}_B \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \bar{s}_B &= \hat{b}_B \\ \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} &= \hat{A}_B^{-1}\hat{b}_B + \hat{A}_B^{-1}\bar{s}_B. \end{aligned} \tag{4.47}$$

For a single variable y_k we get

$$\begin{aligned} \bar{y}_k &= (\hat{A}_B^{-1})_k \hat{b}_B + (\hat{A}_B^{-1})_k \bar{s}_B \\ &= \bar{a}_{k0} + \sum_{i\in B} (-\bar{a}_{ki})\bar{s}_i, \end{aligned} \tag{4.48}$$

where $(\hat{A}_B^{-1})_k$ denotes the k-th row of $\hat{A}_B^{-1}$ and we apply notation (4.25).

Now we introduce a simple disjunctive cut, see e.g., Cornuejols [40].

Definition 4.5. *Let $(\bar{x}, \bar{y})$ be an optimal, basic solution of the continuous relaxation (4.15) with basis B. Let $\bar{y}_k$ be fractional, i.e., $\bar{y}_k \notin \mathbb{N}$. Then the simple disjunctive cut induced by $(\bar{x}, \bar{y})$ is defined by*

$$\pi^T \bar{s}_B \geq \pi_0, \tag{4.49}$$

with $\pi \in \mathbb{R}^n$ and $\pi_0 \in \mathbb{R}$ satisfying

$$\begin{aligned} \pi_0 &:= (\bar{y}_k - \lfloor \bar{y}_k \rfloor)(\lceil \bar{y}_k \rceil - \bar{y}_k), \\ \pi_i^1 &:= (\bar{y}_k - \lceil \bar{y}_k \rceil)(\hat{A}_B^{-1})_{ki}, && \forall i \in B \\ \pi_i^2 &:= (\bar{y}_k - \lfloor \bar{y}_k \rfloor)(\hat{A}_B^{-1})_{ki}, && \forall i \in B, \\ \pi_i &:= \max\{\pi_i^1, \pi_i^2\}, && \forall i \in B \end{aligned} \tag{4.50}$$

where $(\hat{A}_B^{-1})_{ki}$ denotes the i-th entry of the k-th row of $\hat{A}_B^{-1}$, which is the inverse of the basis matrix $\hat{A}_B$.

Note, that the simple disjunctive cut is associated with the disjunction $y_k \geq \lceil \bar{y}_k \rceil \vee y_k \leq \lfloor \bar{y}_k \rfloor$.

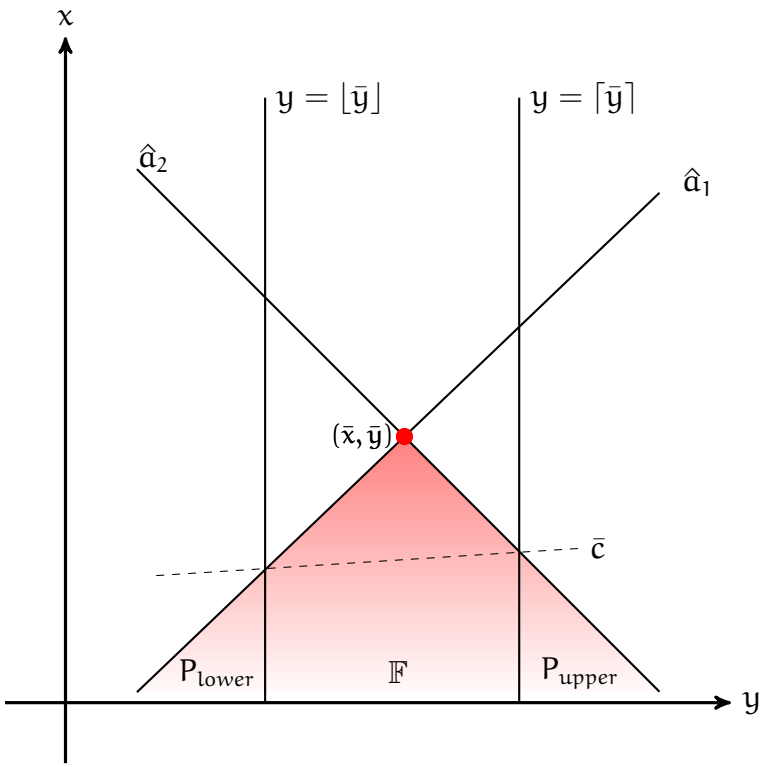

Fig. 4.1: Simple Disjunctive Cutting Plane

Figure 4.1 shows the dashed simple disjunctive cut, denoted by $\bar{c}$, that is determined by the basic solution $(\bar{x}, \bar{y})$ of the continuous relaxation (4.15) and the disjunction on y. The active constraints determining the basic solution $(\bar{x}, \bar{y})$ are here denoted by $\hat{a}_1$ and $\hat{a}_2$ and the feasible region is denoted by $\mathbb{F}$. As described in Section 4.2, the disjunction (4.27) also determines the two polyhedra P_{lower} and P_{upper} defined by (4.31) and (4.32) respectively.

4.4 The Simple Disjunctive Cut and the CGLP

In this section we establish the connection between the simple disjunctive cut (4.49) and the cut generating linear program (4.45). This correlation can be exploited to obtain the efficient cut generation procedure proposed by Perregaard [86], which is reviewed in the next section.

First we will review some results stated by Perregaard [86] and Balas [11].

Lemma 4.1. *The values for both variables u_0 and v_0 in any feasible, basic solution of the CGLP$_k$ given by (4.45), that yields a cutting plane, i.e., a valid linear inequality violated by the optimal solution $(\bar{x}, \bar{y}) \in \mathbb{R}^n$ of the continuous relaxation (4.15), are strictly positive.*

Proof. See Perregaard [86]. □

The following Lemma states some important properties of certain feasible, basic solutions of the CGLP_k, which will be exploited for deriving the efficient cut-generator proposed by Perregaard [86].

Lemma 4.2. *Let* $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ *be a feasible, basic solution of the CGLP*$_k$ (4.45) *with* $\bar{u}_0 > 0$, $\bar{v}_0 > 0$ *and* $(\bar{a}_c, \bar{b}_c)$ *basic. Let the index sets of the basic components of* $\bar{u}$ *and* $\bar{v}$ *be denoted by* M_1 *and* M_2*, respectively.*

Then the following properties are satisfied:

$$M_1 \cap M_2 = \emptyset, \tag{4.51}$$

$$|M_1 \cup M_2| = n. \tag{4.52}$$

Furthermore, the $n \times n$ *sub-matrix* $\hat{A}_{M_1 \cup M_2}$ *is nonsingular. Note, that* $\hat{A}_{M_1 \cup M_2}$ *is the matrix consisting of those constraints that correspond to the basic components of* $\bar{u}$ *and* $\bar{v}$*.*

Proof. See Perregaard [86]. □

The subsequent theorem, which was established by Balas and Perregaard [17] relates a certain feasible, basic solution of the CGLP_k to the simple disjunctive cut $\pi^T s_B \geq \pi_0$ given by (4.49).

Theorem 4.2. *Let* $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ *be a feasible, basic solution of the CGLP*$_k$ (4.45) *with* $\bar{u}_0,\ \bar{v}_0 > 0$ *and all components of* $(\bar{a}_c, \bar{b}_c)$ *basic. Let the index sets of the basic components of* $\bar{u}$ *and* $\bar{v}$ *be denoted by* M_1 *and* M_2*, respectively. Let* $(\bar{x}, \bar{y})$ *be the optimal solution of the continuous relaxation given by* (4.15)*. Let* $\pi^T s_B \geq \pi_0$ *be the simple disjunctive cut corresponding to the disjunction* $y_k \leq \lfloor \bar{y}_k \rfloor \vee y_k \geq \lceil \bar{y}_k \rceil$ *with* $B := M_1 \cup M_2$*, see* (4.49)*.*

Then $\pi^T s_B \geq \pi_0$ *is equivalent to* $\bar{a}_c^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \bar{b}_c$.

Theorem 4.2 establishes equivalence between a simple disjunctive cut and the cut given by a certain feasible, basic solution of the CGLP_k. The proof as given by Perregaard [86], is based on the subsequent assignment, where the cut coefficients of the simple disjunctive cut, i.e., π_0, π^1, π^2 and π are assigned to the variables $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ of CGLP_k:

$$\begin{aligned}
\bar{a}_c &:= \left(\hat{A}_B^T \pi\right) \theta, \\
\bar{u}_B &:= \left(\pi - \pi^1\right) \theta, \\
\bar{u}_0 &:= \left(\lceil \bar{y}_k \rceil - \bar{y}_k\right) \theta, \\
\bar{b}_c &:= \left(\pi_0 + \hat{b}_B^T \pi\right) \theta, \\
\bar{v}_B &:= \left(\pi - \pi^2\right) \theta, \\
\bar{v}_0 &:= \left(\bar{y}_k - \lfloor \bar{y}_k \rfloor\right) \theta.
\end{aligned} \tag{4.53}$$

Note, that due to the normalization constraint (4.46) a scaling factor $\theta \neq 0$ needs to be introduced.

Proof. See Perregaard [86]. □

Theorem 4.2 can be interpreted in the following way:

Consider any, not necessarily feasible, basis and the corresponding basic solution $(\hat{x}, \hat{y})$ of the continuous relaxation (4.15). Let $(\hat{x}, \hat{y})$ be located in the disjunction determined by the optimal, basic solution $(\bar{x}, \bar{y})$ for variable y_k, i.e., $\lfloor \bar{y}_k \rfloor \leq \hat{y}_k \leq \lceil \bar{y}_k \rceil$ holds. Then this basis induces a feasible, basic solution of the CGLP_k via the corresponding simple disjunctive cut according to assignment (4.53).

The following definition intuitively eases the classification of feasible solutions of the CGLP_k.

Definition 4.6. *A feasible, basic solution* $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ *of the CGLP*$_k$ (4.45) *is called a cut-generating, feasible, basic solution, if* $\bar{u}_0$ *and* $\bar{v}_0$ *are both strictly positive and all components of* $(\bar{a}_c, \bar{b}_c)$ *are basic, i.e., contained in the basis, and the objective value of the CGLP*$_k$ *is strictly negative.*

A feasible, basic solution $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ *of the CGLP*$_k$ (4.45) *is called a non-trivial, feasible, basic solution, if it is obtained from a basis* B *of LP* (4.15) *via the induced simple disjunctive cut* (4.49) *and assignment* (4.53).

A feasible solution $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ *of CGLP*$_k$ *is called a trivial, feasible solution of the CGLP*$_k$*, if*

$$\bar{u}_0 \bar{v}_0 = 0 \tag{4.54}$$

holds.

Note, that due to Lemma 4.1 only a non-trivial, feasible, basic solution can be a cut-generating, feasible, basic solution of the CGLP_k. A trivial, feasible solution of CGLP_k yields no cut, since the coefficients $(\bar{a}_c, \bar{b}_c)$ of the corresponding inequality are determined by a non-negative linear combination of some of the original constraints within the CGLP_k, see Perregaard [86] for further details. This observation is stated in the subsequent corollary.

Corollary 4.2. *A cut-generating, feasible, basic solution of the CGLP*$_k$ *is always a non-trivial, feasible, basic solution. A trivial, feasible solution of the CGLP*$_k$ *cannot be a cut-generating, feasible, basic solution.*

Proof. Apply Lemma 4.1 and Definition 4.6. □

4.5 An Efficient Cut Generation Procedure for Disjunctive Cutting Planes

Exploiting the previously reviewed theory related to disjunctive programming, Perregaard proposed an efficient cut generation procedure for disjunctive cutting planes [86]. It is essentially based on the equivalence of the simple disjunctive cut (4.49) and a non-trivial, feasible, basic solution of the $CGLP_k$ (4.45). Furthermore, this cut generation procedure solves the cut generating linear program implicitly in the original problem dimension. Therefore, it yields substantial advantages, since the computational effort for constructing disjunctive cutting planes is significantly decreased compared to solving the high-dimensional, highly degenerate $CGLP_k$. Apart from working within the original problem dimensions, the computation time is also improved by the reduction of degeneracy. The reason is that the partition of B into M_1 and M_2 is determined automatically by the sign of the corresponding entry in the k-th row of $(\hat{A}_B^{-1})$, see Theorem 4.2.

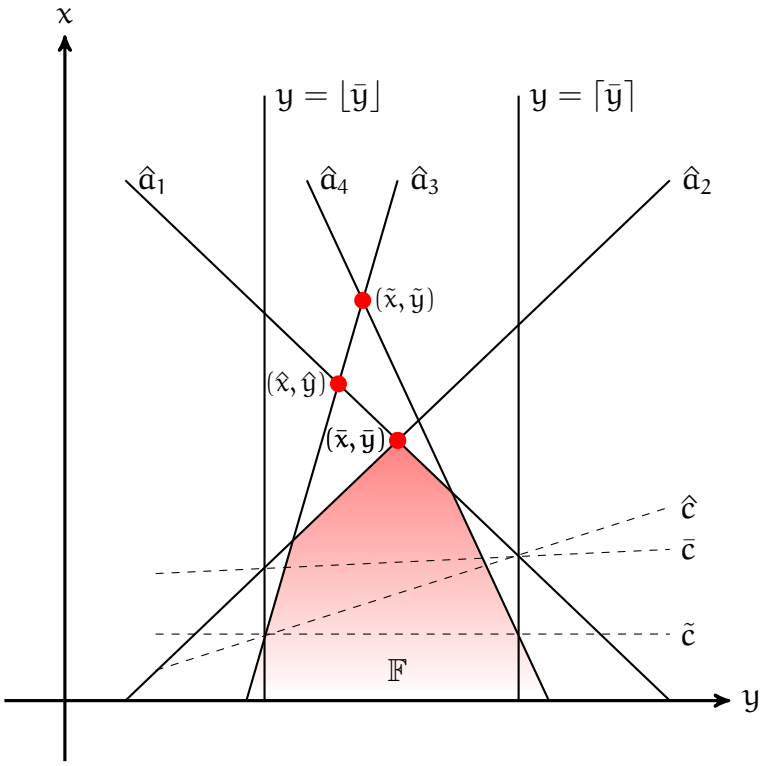

Fig. 4.2: Disjunctive Cutting Plane

The proposed method starts with the simple disjunctive cut induced by the basic solution $(\bar{x}, \bar{y})$ of the continuous relaxation of MILP (4.26). Due to Theorem 4.2 this simple disjunctive cut is equivalent to a cut-generating, feasible, basic solution of the cut generating linear program. Analogue to the simplex algorithm, Perregaard suggests to find an adjacent, cut-generating, feasible, basic solution possessing a lower

objective value for the $CGLP_k$ than the current cut-generating, feasible, basic solution. The new cut-generating, feasible, basic solution also corresponds to a simple disjunctive cut. These two simple disjunctive cuts are closely related, since they only differ in a single constraint, i.e., one basic constraint is exchanged in the corresponding basis, see Definition 4.3. A reduction of the objective function value means, that the adjacent cut has to be stronger than the current one, i.e., it is violated more by the solution $(\bar{x}, \bar{y})$ of the continuous relaxation of MILP (4.26).

Figure 4.2 illustrates the procedure. It starts at the solution $(\bar{x}, \bar{y})$ of the continuous relaxation (4.12) that is determined by the active constraints $\hat{a}_1$ and $\hat{a}_2$. Since $\bar{y}$ is fractional, $(\bar{x}, \bar{y})$ implies the simple disjunctive cut $\bar{c}$. Considering inactive constraints we replace $\hat{a}_2$ by $\hat{a}_3$, which yields the basic solution $(\hat{x}, \hat{y})$ and the corresponding simple disjunctive cut $\hat{c}$. Note, that $(\hat{x}, \hat{y})$ is infeasible, since constraint $\hat{a}_2$ is not satisfied. Obviously, $\hat{c}$ is stronger than $\bar{c}$, since it is violated more by the solution $(\bar{x}, \bar{y})$ of the continuous relaxation. If the constraint $\hat{a}_1$ is replaced by constraint $\hat{a}_4$, the simple disjunctive cut $\tilde{c}$ is induced by the corresponding basic solution $(\tilde{x}, \tilde{y})$. It is equivalent to the disjunctive cutting plane determined by the $CGLP_k$, set up by the constraints $\hat{a}_1$, $\hat{a}_2$, $\hat{a}_3$ and $\hat{a}_4$ defining the feasible region $\mathbb{F}$.

To apply this procedure, Perregaard derives a criterion, that determines whether a constraint $(\hat{a}_i, \hat{b}_i)$ corresponding to variables u_i and v_i, that are not contained in the current basis of the $CGLP_k$, should be included in the basis of the original problem via a basis exchange to obtain a stronger cutting plane, or not. The subsequent theorem stated by Perregaard [86] gives formulas for calculating reduced costs for every non-basic variable u_i and v_i, i.e., every constraint that is not included in the basis B determining the current cut, which is derived by assignment (4.53) from the simple disjunctive cut induced by B.

Theorem 4.3. *Consider a non-trivial, feasible, basic solution of CGLP$_k$* (4.45) *given by* $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$*, i.e.,* $\bar{u}_0,\ \bar{v}_0 > 0$ *and* $(\bar{a}_c, \bar{b}_c)$ *basic. Assume, that the basic components of* $\bar{u}$ *and* $\bar{v}$ *are indexed by* M_1 *and* M_2*, respectively, and denote the basis formed by the constraints corresponding to* M_1 *and* M_2 *by* B*, i.e.,* $B := M_1 \cup M_2$*. Let* $\bar{s}$ *correspond to the slack variables* (5.25) *introduced in the system* $\hat{A}\begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}$ *at the optimal solution* $(\bar{x}, \bar{y})$ *of the continuous relaxation of MILP* (4.26)*. The reduced costs for* $\bar{u}_i$ *and* $\bar{v}_i$*,* $i \notin B \cup \{k\}$ *are*

$$\begin{aligned} r_{\bar{u}_i} &:= -\sigma - \tau - \bar{a}_{i0}(\bar{y}_k - \lceil \bar{y}_k \rceil) \\ r_{\bar{v}_i} &:= -\sigma + \tau + \bar{a}_{i0}(\bar{y}_k - \lceil \bar{y}_k \rceil) + \bar{s}_i \end{aligned} \tag{4.55}$$

with

$$\sigma := \frac{\sum\limits_{j \in M_2} \bar{a}_{kj}\bar{s}_j + (\bar{a}_{k0} - \lfloor \bar{y}_k \rfloor)(\bar{y}_k - \lceil \bar{y}_k \rceil)}{1 + \sum\limits_{j \in B} |\bar{a}_{kj}|} \tag{4.56}$$

$$\tau := \sigma \sum_{j \in M_1} \bar{a}_{ij} - \sigma \sum_{j \in M_2} \bar{a}_{ij} + \sum_{j \in M_2} \bar{a}_{ij}\bar{s}_j. \tag{4.57}$$

and $\bar{a}_{k0}$, $\bar{a}_{i0}$, $\bar{a}_{kj}$ *and* $\bar{a}_{ij}$ *defined in* (4.25).

We review the most important parts of the proof and extend it for general integer variables, since the understanding is important for the generalization for MIQPs in Chapter 5. For details on some reformulations see Perregaard [86].

Proof. In the first part of the proof, we exploit, that $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ is a feasible, basic solution of the $CGLP_k$ to derive formulas for the variables $\bar{u}_j$ with $j \in M_1$ and $\bar{v}_j$ with $j \in M_2$ as well as $\bar{u}_0$ and $\bar{v}_0$. Afterwards, these formulas are feed into the objective function of the $CGLP_k$ together with one pair $\bar{u}_i$ and $\bar{v}_i$ with $i \notin B$ of non-basic variables. This yields the desired expressions for the reduced costs of $\bar{u}_i$ and $\bar{v}_i$.

Since $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ is a feasible basic solution of the $CGLP_k$ (4.45), we obtain

$$\hat{A}_{M_1}^T \bar{u}_{M_1} - \bar{u}_0 e_k = \hat{A}_{M_2}^T \bar{v}_{M_2} + \bar{v}_0 e_k \tag{4.58}$$

from

$$\bar{a}_c - \hat{A}^T \bar{u} + \bar{u}_0 e_k = 0 \tag{4.59}$$

$$\bar{a}_c - \hat{A}^T \bar{v} - \bar{v}_0 e_k = 0 \tag{4.60}$$

by eliminating $\bar{a}_c$. Note, that we denote the index sets of the basic variables $\bar{u}$ and $\bar{v}$ by M_1 and M_2, respectively. Therefore, the index set of the basis B is partitioned into M_1 and M_2. Furthermore, we derive

$$\hat{b}_{M_1}^T \bar{u}_{M_1} - \bar{u}_0 \lfloor \bar{y}_k \rfloor = \hat{b}_{M_2}^T \bar{v}_{M_2} + \bar{v}_0 \lceil \bar{y}_k \rceil \tag{4.61}$$

from

$$-\bar{b}_c + \hat{b}^T \bar{u} - \bar{u}_0 \lfloor \bar{y}_k \rfloor = 0 \tag{4.62}$$

$$-\bar{b}_c + \hat{b}^T \bar{v} + \bar{v}_0 \lceil \bar{y}_k \rceil = 0 \tag{4.63}$$

by eliminating $\bar{b}_c$, see again (4.45). Moreover, the normalization constraint (4.46) is satisfied, which gives

$$\sum_{j \in M_1} \bar{u}_j + \bar{u}_0 + \sum_{j \in M_2} \bar{v}_j + \bar{v}_0 = 1, \tag{4.64}$$

Note, that all non-basic variables $\bar{u}_i$, $\bar{v}_i$ can be eliminated, since $\bar{u}_i = 0$, $\forall i \notin M_1$ and $\bar{v}_i = 0$, $\forall i \notin M_2$ holds. Furthermore, we introduce one arbitrary non-basic variable $\bar{u}_i$ and one non-basic variable $\bar{v}_i$ with $i \notin B$, which are both zero. In order to derive reduced costs for $\bar{u}_i$ and $\bar{v}_i$, we restate the equations (4.58) and (4.61) using $\hat{A}_B := \begin{bmatrix} \hat{A}_{M_1} \\ \hat{A}_{M_2} \end{bmatrix}$

$$\hat{A}_B^T \begin{pmatrix} \bar{u}_{M_1} \\ -\bar{v}_{M_2} \end{pmatrix} + (\bar{u}_i - \bar{v}_i)\hat{a}_i = (\bar{u}_0 + \bar{v}_0) e_k \tag{4.65}$$

$$\hat{b}_B^T \begin{pmatrix} \bar{u}_{M_1} \\ -\bar{v}_{M_2} \end{pmatrix} + (\bar{u}_i - \bar{v}_i)\hat{b}_i = (\bar{u}_0 + \bar{v}_0) \lfloor \bar{y}_k \rfloor + \bar{v}_0. \tag{4.66}$$

Since $\hat{A}_B$ is invertible due to Lemma 4.2, we obtain from (4.65)

$$\begin{pmatrix} \bar{u}_{M_1} \\ -\bar{v}_{M_2} \end{pmatrix} = \hat{A}_B^{-T}(\bar{u}_0 + \bar{v}_0)e_k - \hat{A}_B^{-T}(\bar{u}_i - \bar{v}_i)\hat{a}_i \tag{4.67}$$

by multiplying with $\hat{A}_B^{-T}$. By replacing $\begin{pmatrix} \bar{u}_{M_1} \\ -\bar{v}_{M_2} \end{pmatrix}$ in (4.66) according to (4.67) we get

$$(\bar{u}_0 + \bar{v}_0)(e_k^T\hat{A}_B^{-1}\hat{b}_B)^T - (\bar{u}_i - \bar{v}_i)((\hat{a}_i^T\hat{A}_B^{-1}\hat{b}_B)^T - \hat{b}_i) = (\bar{u}_0 + \bar{v}_0)\lfloor \bar{y}_k \rfloor + \bar{v}_0 \tag{4.68}$$

after some reformulations. For simplification we apply notation (4.25). Therefore, we obtain the following conditions for $\bar{u}_j$, $j \in M_1$, $\bar{v}_j$, $j \in M_2$ and $\bar{v}_0$ from equations (4.67) and (4.68)

$$\begin{aligned} \bar{u}_j &= -(\bar{u}_0 + \bar{v}_0)\bar{a}_{kj} + (\bar{u}_i - \bar{v}_i)\bar{a}_{ij}, && j \in M_1 \\ \bar{v}_j &= (\bar{u}_0 + \bar{v}_0)\bar{a}_{kj} - (\bar{u}_i - \bar{v}_i)\bar{a}_{ij}, && j \in M_2 \\ \bar{v}_0 &= (\bar{u}_0 + \bar{v}_0)(\bar{a}_{k0} - \lfloor \bar{y}_k \rfloor) - (\bar{u}_i - \bar{v}_i)\bar{a}_{i0}. \end{aligned} \tag{4.69}$$

We can insert these expressions into the normalization constraint (4.64) extended by $\bar{u}_i(=0)$ and $\bar{v}_i(=0)$ to get

$$\begin{aligned} 1 &= \sum_{j\in M_1} \bar{u}_j + \bar{u}_0 + \sum_{j\in M_2} \bar{v}_j + \bar{v}_0 + \bar{u}_i + \bar{v}_i \\ &= (\bar{u}_0 + \bar{v}_0)\left(-\sum_{j\in M_1} \bar{a}_{kj} + \sum_{j\in M_2} \bar{a}_{kj} + 1\right) \\ &\quad +(\bar{u}_i - \bar{v}_i)\left(\sum_{j\in M_1} \bar{a}_{ij} - \sum_{j\in M_2} \bar{a}_{ij}\right) + \bar{u}_i + \bar{v}_i. \end{aligned} \tag{4.70}$$

Since $(\bar{a}_c, \bar{b}_c)$ is equivalent to a simple disjunctive cut we know that all basic components $(\bar{u}_{M_1}, \bar{v}_{M_2})$ are greater or equal to zero, see (4.53). Therefore equations (4.69) yield

$$-\sum_{j\in M_1} \underbrace{\bar{a}_{kj}}_{\leq 0} + \sum_{j\in M_2} \underbrace{\bar{a}_{kj}}_{\geq 0} = \sum_{j\in B} |\bar{a}_{kj}| \tag{4.71}$$

since $\bar{u}_j, \bar{v}_j, \bar{v}_0 \geq 0$. Therefore, we obtain for $\bar{u}_0 + \bar{v}_0$ from (4.70):

$$\bar{u}_0 + \bar{v}_0 = \frac{1 - (\bar{u}_i - \bar{v}_i)\left(\sum_{j\in M_1} \bar{a}_{ij} - \sum_{j\in M_2} \bar{a}_{ij}\right) - \bar{u}_i - \bar{v}_i}{1 + \sum_{j\in B} |\bar{a}_{kj}|}. \tag{4.72}$$

The reduced costs for $\bar{u}_i$ and $\bar{v}_i$ with $i \notin B$ determine how the objective value changes, if the corresponding variables $\bar{u}_i$ or $\bar{v}_i$ is increased. As a consequence, we have to express the objective function of the CGLP_k given by

$$\bar{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \bar{b}_c \tag{4.73}$$

in terms of $\bar{u}_i$ and $\bar{v}_i$. Applying (4.60) and (4.63) yields

$$\begin{aligned} \bar{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \bar{b}_c &= \hat{A}_{M_2}^T \bar{v}_{M_2} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{a}_i^T \bar{v}_i \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \bar{v}_0 e_k \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \\ &\quad - \hat{b}_{M_2}^T \bar{v}_{M_2} - \hat{b}_i \bar{v}_i - \bar{v}_0 \lceil \bar{y}_k \rceil \\ &= \bar{s}_{M_2}^T \bar{v}_{M_2} + \bar{s}_i \bar{v}_i + \bar{v}_0 (\bar{y}_k - \lceil \bar{y}_k \rceil), \end{aligned} \tag{4.74}$$

where

$$\bar{s}_{M_2} := \hat{A}_{M_2} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_{M_2}, \tag{4.75}$$

$$\bar{s}_i := \hat{a}_i \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_i \tag{4.76}$$

holds. We can now substitute $\bar{v}_{M_2}$ and $\bar{v}_0$ by the expressions derived in (4.69). This gives

$$\begin{aligned} \bar{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \bar{b}_c &= (\bar{u}_0 + \bar{v}_0) \left(\sum_{j \in M_2} \bar{a}_{kj} \bar{s}_j \right) - (\bar{u}_i - \bar{v}_i) \left(\sum_{j \in M_2} \bar{a}_{ij} \bar{s}_j \right) + \bar{v}_i \bar{s}_i \\ &\quad + ((\bar{u}_0 + \bar{v}_0)(\bar{a}_{k0} - \lfloor \bar{y}_k \rfloor) - (\bar{u}_i - \bar{v}_i) \bar{a}_{i0}) (\bar{y}_k - \lceil \bar{y}_k \rceil) \\ &= (\bar{u}_0 + \bar{v}_0) \left(\sum_{j \in M_2} \bar{a}_{kj} \bar{s}_j + (\bar{a}_{k0} - \lfloor \bar{y}_k \rfloor)(\bar{y}_k - \lceil \bar{y}_k \rceil) \right) \\ &\quad + (\bar{u}_i - \bar{v}_i) \left(- \sum_{j \in M_2} \bar{a}_{ij} \bar{s}_j - \bar{a}_{i0} (\bar{y}_k - \lceil \bar{y}_k \rceil) \right) + \bar{v}_i \bar{s}_i. \end{aligned} \tag{4.77}$$

After replacing $\bar{u}_0 + \bar{v}_0$ according to (4.72) and introducing

$$\sigma := \frac{\sum_{j \in M_2} \bar{a}_{kj} \bar{s}_j + (\bar{a}_{k0} - \lfloor \bar{y}_k \rfloor)(\bar{y}_k - \lceil \bar{y}_k \rceil)}{1 + \sum_{j \in B} |\bar{a}_{kj}|} \tag{4.78}$$

we obtain

$$\begin{aligned} \bar{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \bar{b}_c &= \sigma - \sigma \bar{u}_i - \sigma \bar{v}_i - (\bar{u}_i - \bar{v}_i) \sigma \left(\sum_{j \in M_1} \bar{a}_{ij} - \sum_{j \in M_2} \bar{a}_{ij} \right) \\ &\quad + (\bar{u}_i - \bar{v}_i) \left(- \sum_{j \in M_2} \bar{a}_{ij} \bar{s}_j - \bar{a}_{i0} (\bar{y}_k - \lceil \bar{y}_k \rceil) \right) + \bar{v}_i \bar{s}_i. \end{aligned} \tag{4.79}$$

Let

$$\tau := \sigma \sum_{j \in M_1} \bar{a}_{ij} - \sigma \sum_{j \in M_2} \bar{a}_{ij} + \sum_{j \in M_2} \bar{a}_{ij} \bar{s}_j, \tag{4.80}$$

then we get for the objective function the following expression defining the reduced costs for $\bar{u}_i$ and $\bar{v}_i$

$$\begin{aligned} \bar{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \bar{b}_c &= \sigma + \bar{u}_i(-\sigma - \tau - \bar{a}_{i0}(\bar{y}_k - \lceil \bar{y}_k \rceil)) \\ &\quad + \bar{v}_i(-\sigma + \tau + \bar{a}_{i0}(\bar{y}_k - \lceil \bar{y}_k \rceil) + \bar{s}_i). \end{aligned} \tag{4.81}$$

This proves the theorem. □

Theorem 4.3 can be used to identify non-basic constraints, that strengthen the current cut, if they are included in the basis. To solve the CGLP_k (4.45) implicitly, constraints, that should be removed from the current basis, need to be identified as well, such that a basis exchange can be executed. The following theorem stated by Perregaard [86] can be applied for this task.

Theorem 4.4. *Let $(\bar{a}_c, \bar{b}_c, \bar{u}, \bar{u}_0, \bar{v}, \bar{v}_0)$ be a non-trivial, feasible, basic solution of the $CGLP_k$. Assume, that the basic components of $\bar{u}$ and $\bar{v}$ are indexed by M_1 and M_2 respectively, with $B := M_1 \cup M_2$. Let $\bar{u}_i$ or $\bar{v}_i$, corresponding to constraint $(\hat{a}_i, \hat{b}_i)$ with $i \notin B$, be a given non-basic variable. Then the cut $(\bar{a}_c, \bar{b}_c)$ given by the current non-trivial, feasible, basic solution of the $CGLP_k$ is improved most, if the basic variable $\bar{u}_{l^*}$ or $\bar{v}_{l^*}$, corresponding to constraint $(\hat{a}_{l^*}, \hat{b}_{l^*})$, with $l^* \in B$ is removed. Index l^* is given by the basic variable, that minimizes $f^+(\gamma_l)$ for $\gamma_l := -\frac{\bar{a}_{kl}}{\bar{a}_{il}} > 0$ and $f^-(\gamma_l)$ for $\gamma_l < 0$. For some γ, $f^+(\gamma)$ and $f^-(\gamma)$ are given by*

$$f^+(\gamma) := \frac{\pi^{\gamma^+} \bar{s}_B - \pi_0^{\gamma^+}}{1 + |\gamma| + \sum_{j \in B} |\bar{a}_{kj} + \gamma \bar{a}_{ij}|} \tag{4.82}$$

$$f^-(\gamma) := \frac{\pi^{\gamma^-} \bar{s}_B - \pi_0^{\gamma^-}}{1 + |\gamma| + \sum_{j \in B} |\bar{a}_{kj} + \gamma \bar{a}_{ij}|}. \tag{4.83}$$

with

$$\begin{aligned} \pi^{\gamma^+} \bar{s}_B - \pi_0^{\gamma^+} &= \sum_{j \in B} (\max\{0, -(\bar{a}_{kj} + \gamma \bar{a}_{ij})\} \bar{s}_j) \\ &\quad + (\lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil - \bar{a}_{k0} - \gamma \bar{a}_{i0})(\lfloor \bar{a}_{k0} + \gamma \bar{a}_{i0} \rfloor - \bar{y}_k) \end{aligned} \tag{4.84}$$

$$\begin{aligned} \pi^{\gamma^-} \bar{s}_B - \pi_0^{\gamma^-} &= \sum_{j \in B} (\max\{\gamma \bar{a}_{ij}, -\bar{a}_{kj}\} \bar{s}_j) \\ &\quad + (\lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil - \bar{a}_{k0} - \gamma \bar{a}_{i0})(\lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil - \bar{y}_k) \\ &\quad + \bar{a}_{k0} - \lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil. \end{aligned} \tag{4.85}$$

based on the notation specified in (4.25).

Again we review the most important parts of the proof and extend it for general integer variables. For further details on some reformulations we refer to Perregaard [86].

Proof. $(\bar{x}, \bar{y})$ is the optimal basic solution of the continuous relaxation of MILP (4.26) with $\bar{y}_k$ fractional. Then the simple disjunctive cut for the composite row of the simplex tableau, obtained by adding the row corresponding to $\bar{y}_k$ and the row corresponding to $\bar{x}_i$ (or $\bar{y}_i$) multiplied by γ, is constructed first. It directly leads to the expressions denoted by $f^+(\gamma)$ and $f^-(\gamma)$, which need to be minimized in order to obtain the strongest simple disjunctive cut.

Assume, that for an constraint i not contained in the basis inducing the current simple disjunctive cut, either $\bar{u}_i$ or $\bar{v}_i$ with $i \notin B$, possesses negative reduced costs (4.55) and should therefore be included in the disjunctive cut derived for $\bar{y}_k$. Consider the simplex tableau representation (4.47) corresponding to $\bar{y}_k$, i.e., row k

$$\bar{y}_k + \sum_{j \in B} \bar{a}_{kj} \bar{s}_j \quad = \quad \bar{a}_{k0} \tag{4.86}$$

and $\bar{x}_i$ (or $\bar{y}_i$), i.e., row i

$$\bar{x}_i + \sum_{j \in B} \bar{a}_{ij} \bar{s}_j \quad = \quad \bar{a}_{i0}, \tag{4.87}$$

applying notation (4.25). The non-trivial, feasible, basic solution of $CGLP_k$ corresponds to the simple disjunctive cut for row k. Since we want to include constraint i with $i \notin B$ in the cut we add row i given by (4.87) multiplied by γ to row k and calculate the simple disjunctive cut corresponding to the composite row:

$$\bar{y}_k + \gamma \bar{x}_i + \sum_{j \in B} (\bar{a}_{kj} + \gamma \bar{a}_{ij}) \bar{s}_j \quad = \quad \bar{a}_{k0} + \gamma \bar{a}_{i0}. \tag{4.88}$$

When constructing the composite row (4.88), we can still choose the multiplier γ. γ has to satisfy property

$$\lfloor \bar{a}_{k0} + \gamma \bar{a}_{i0} \rfloor \quad = \quad \lfloor \bar{y}_k \rfloor. \tag{4.89}$$

to ensure that the corresponding simple disjunctive cut is valid, i.e., the new basic solution is located in the disjunction determined by $\bar{y}_k$. In general, choosing $\gamma := -\dfrac{\bar{a}_{kl}}{\bar{a}_{il}}$ with $\bar{a}_{il} \neq 0$ has the effect, that the basic variable $\bar{u}_l$ or $\bar{v}_l$, corresponding to constraint $(\hat{a}_l, \hat{b}_l)$, becomes non-basic, i.e., $\bar{u}_l = \bar{v}_l = 0$, while either $\bar{u}_i$ or $\bar{v}_i$ becomes basic instead.

Therefore, we have to determine the column $l \in B$ with the corresponding $\gamma := -\dfrac{\bar{a}_{kl}}{\bar{a}_{il}}$ such that the new cut $\pi^\gamma s_{\hat{B}} \geq \pi_0^\gamma$ with $\hat{B} := B \backslash \{l\} \cup \{i\}$ determined by the composite row (4.88) is as strong as possible. This is equivalent to minimizing $\pi^\gamma s_{\hat{B}} -$

π_0^γ. Therefore we can apply the formulas (4.49) and (4.50) treating the term $\gamma\bar{x}_i$ similar to $\bar{s}_B$ to obtain the following coefficients for the new cut:

$$\begin{aligned}
\pi_0^\gamma &:= (\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})(\bar{a}_{k0} + \gamma\bar{a}_{i0} - \lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0}\rfloor)\\
(\pi_i^1)^\gamma &:= (\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})\gamma\\
(\pi_i^2)^\gamma &:= -(\bar{a}_{k0} + \gamma\bar{a}_{i0} - \lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0}\rfloor)\gamma\\
\pi_i^\gamma &:= \max\{(\pi_i^1)^\gamma, (\pi_i^2)^\gamma\} \qquad (4.90)\\
(\pi_j^1)^\gamma &:= (\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})(\bar{a}_{kj} + \gamma\bar{a}_{ij}), \qquad j \in B,\\
(\pi_j^2)^\gamma &:= -(\bar{a}_{k0} + \gamma\bar{a}_{i0} - \lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0}\rfloor)(\bar{a}_{kj} + \gamma\bar{a}_{ij}), \qquad j \in B,\\
\pi_j^\gamma &:= \max\{(\pi_j^1)^\gamma, (\pi_j^2)^\gamma\}, \qquad j \in B.
\end{aligned}$$

These coefficients correspond to the simple disjunctive cut $\pi_i^\gamma\bar{x}_i + \pi^\gamma\bar{s}_B \geq \pi_0^\gamma$. We can eliminate $\bar{x}_i$ by subtracting π_i^γ times row i (4.87). Obviously, the result depends on the sign of γ, since the coefficient π_i^γ of $\bar{x}_i$ depends on the sign of γ.

For $\gamma > 0$ we obtain for all $j \in B$ by subtracting γx_i, where x_i is defined by (4.87), the coefficients

$$\begin{aligned}
\pi_j^{\gamma^+} &:= \pi_j^\gamma - (\pi_i^1)^\gamma\bar{a}_{ij}\\
&= \max\{(\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})(\bar{a}_{kj} + \gamma\bar{a}_{ij}),\\
&\quad -(\bar{a}_{k0} + \gamma\bar{a}_{i0} - \lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0}\rfloor)(\bar{a}_{kj} + \gamma\bar{a}_{ij})\} - (\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})\gamma\bar{a}_{ij}\\
&= (\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})\bar{a}_{kj} + \max\{0, -(\bar{a}_{kj} + \gamma\bar{a}_{ij})\}.
\end{aligned}$$

Furthermore, the right hand side of the cut is given by

$$\begin{aligned}
\pi_0^{\gamma^+} &:= \pi_0^\gamma - (\pi_i^1)^\gamma\bar{a}_{i0}\\
&= (\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})(\bar{a}_{k0} - \lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0}\rfloor).
\end{aligned}$$

For $\gamma < 0$ the coefficients are

$$\begin{aligned}
\pi_j^{\gamma^-} &:= \pi_j^\gamma - (\pi_i^2)^\gamma\bar{a}_{ij}\\
&= \max\{(\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})(\bar{a}_{kj} + \gamma\bar{a}_{ij}),\\
&\quad -(\bar{a}_{k0} + \gamma\bar{a}_{i0} - \lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0}\rfloor)(\bar{a}_{kj} + \gamma\bar{a}_{ij})\} + (\bar{a}_{k0} + \gamma\bar{a}_{i0} - \lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0}\rfloor)\gamma\bar{a}_{ij}\\
&= (\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0} - \gamma\bar{a}_{i0})\bar{a}_{kj} + \max\{\gamma\bar{a}_{ij}, -\bar{a}_{kj}\},
\end{aligned}$$

while the right hand side is given by

$$\begin{aligned}
\pi_0^{\gamma^-} &:= \pi_0^\gamma - (\pi_i^2)^\gamma\bar{a}_{i0}\\
&= (\bar{a}_{k0} + \gamma\bar{a}_{i0} - \lfloor \bar{a}_{k0} + \gamma\bar{a}_{i0}\rfloor)(\lceil \bar{a}_{k0} + \gamma\bar{a}_{i0}\rceil - \bar{a}_{k0}).
\end{aligned}$$

See also Perregaard [86] for details. The simple disjunctive cut of the composite row (4.88) is again equivalent to a non-trivial, feasible, basic solution $(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ of $CGLP_k$ with the following properties

$$\begin{aligned} \hat{u}_i + \hat{v}_i &= (\pi_i^\gamma - (\pi_i^1)^\gamma) + (\pi_i^\gamma - (\pi_i^2)^\gamma) && (4.91)\\ &= |(\pi_i^1)^\gamma - (\pi_i^2)^\gamma| = |\gamma| \\ \hat{u}_j + \hat{v}_j &= (\pi_j^\gamma - (\pi_j^1)^\gamma) + (\pi_j^\gamma - (\pi_j^2)^\gamma) && (4.92)\\ &= |(\pi_j^1)^\gamma - (\pi_j^2)^\gamma| = |\bar{a}_{kj} + \gamma \bar{a}_{ij}| \quad \forall j \in B \\ \hat{u}_0 + \hat{v}_0 &= (\lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil - \bar{a}_{k0} - \gamma \bar{a}_{i0}) && (4.93)\\ &+ (\bar{a}_{k0} + \gamma \bar{a}_{i0} - \lfloor \bar{a}_{k0} + \gamma \bar{a}_{i0} \rfloor) = 1, \end{aligned}$$

satisfying all constraints of $CGLP_k$ except the normalization constraint (4.46), since

$$\sum_{j \in B} \hat{u}_j + \hat{u}_i + \hat{u}_0 + \sum_{j \in B} \hat{v}_j + \hat{v}_i + \hat{v}_0 = 1 + |\gamma| + \sum_{j \in B} |\bar{a}_{kj} + \gamma \bar{a}_{ij}|. \quad (4.94)$$

Scaling the cut with $\dfrac{1}{1 + |\gamma| + \sum_{j \in B} |\bar{a}_{kj} + \gamma \bar{a}_{ij}|}$ leads to a feasible, basic solution of $CGLP_k$ satisfying also the normalization constraint (4.46). Furthermore,

$$\hat{u}_l + \hat{v}_l = |\bar{a}_{kl} + \gamma \bar{a}_{il}| = 0 \quad (4.95)$$

holds, since $\gamma = -\dfrac{\bar{a}_{kl}}{a_{il}}$, where l is the chosen basic variable to be eliminated. Therefore the strongest cut is obtained by eliminating the basic variable $\bar{u}_l$ or $\bar{v}_l$ with $l \in B$, which minimizes the following functions

$$f^+(\gamma) := \frac{\pi^{\gamma^+} \bar{s}_B - \pi_0^{\gamma^+}}{1 + |\gamma| + \sum_{j \in B} |\bar{a}_{kj} + \gamma \bar{a}_{ij}|} \quad (4.96)$$

$$f^-(\gamma) := \frac{\pi^{\gamma^-} \bar{s}_B - \pi_0^{\gamma^-}}{1 + |\gamma| + \sum_{j \in B} |\bar{a}_{kj} + \gamma \bar{a}_{ij}|}. \quad (4.97)$$

We can perform some reformulations to obtain the following expressions applying notation (4.25):

$$\begin{aligned} \pi^{\gamma^+} \bar{s}_B - \pi_0^{\gamma^+} &= \sum_{j \in B} (\max\{0, -(\bar{a}_{kj} + \gamma \bar{a}_{ij})\} \bar{s}_j) && (4.98)\\ &+ (\lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil - \bar{a}_{k0} - \gamma \bar{a}_{i0})(\lfloor \bar{a}_{k0} + \gamma \bar{a}_{i0} \rfloor - \bar{y}_k) \\ \pi^{\gamma^-} \bar{s}_B - \pi_0^{\gamma^-} &= \sum_{j \in B} (\max\{\gamma \bar{a}_{ij}, -\bar{a}_{kj}\} \bar{s}_j) && (4.99)\\ &+ (\lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil - \bar{a}_{k0} - \gamma \bar{a}_{i0})(\lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil - \bar{y}_k) \\ &+ \bar{a}_{k0} - \lceil \bar{a}_{k0} + \gamma \bar{a}_{i0} \rceil. \end{aligned}$$

This proves the theorem. □

Now we can formulate the algorithm proposed by Perregaard [86] to solve CGLP_k implicitly.

Algorithm 4.1. *1. Solve the continuous relaxation* (4.15) *of MILP* (4.26)*. Let* $(\bar{x}, \bar{y})$ *be an optimal, basic solution with* $\lfloor \bar{y}_k \rfloor < \bar{y}_k < \lceil \bar{y}_k \rceil$*, i.e.,* $\bar{y}_k$ *is fractional.*

2. Let B *be the index set of the current basis. Compute the reduced costs given by* (4.55) *for each constraint* $i \notin B,\ i \neq k$*, i.e., for* u_i *and* v_i*. The sets* M_1 *and* M_2 *are given by*

$$M_1 := \{j \in B : \bar{a}_{kj} < 0 \vee (\bar{a}_{kj} = 0 \wedge \bar{a}_{ij} > 0)\}$$

and

$$M_2 := B \backslash M_1$$

for r_{u_i}*. For determining* r_{v_i}*,* M_1 *and* M_2 *are given by*

$$M_1 := \{j \in B : \bar{a}_{kj} < 0 \vee (\bar{a}_{kj} = 0 \wedge \bar{a}_{ij} < 0)\}$$

and

$$M_2 := B \backslash M_1$$

3. Let i^* *be a constraint with* $r_{u_{i^*}} < 0$ *or* $r_{v_{i^*}} < 0$*.*

If i^* *does not exist,* **then GOTO** *Step 7.*

4. Identify the most improving pivot column j_* *in row* i^* *by minimizing* $f^+(\gamma_j)$ *over all* $j \in B$ *with* $\gamma > 0$ *and* $f^-(\gamma_j)$ *over all* $j \in B$ *with* $\gamma_j < 0$*, where* γ_j *is defined by* $\gamma_j := -\dfrac{\bar{a}_{kj}}{\bar{a}_{i^*j}}$*. Choose the smaller of both values.*

5. Pivot on $\bar{a}_{i^*j_*}$*, i.e., replace basic constraint* j_* *with non-basic constraint* i^**.*

6. **GOTO** *Step 2.*

7. Perturbation of row k (4.86)*:*

If *row* k *has no zero entries,* **then STOP**.

Else *perturb row* k *by replacing every zero entry by* ε^t *for some small* $\varepsilon > 0$ *and* $t = 1, 2 \ldots$. **GOTO** *Step 2.*

5. DISJUNCTIVE CUTTING PLANES FOR NON-BASIC SOLUTIONS

In this chapter we propose efficient cut generation methods for disjunctive cutting planes for non-basic solutions. This is one of the main results of this thesis. Based on available theory presented in Chapter 4, we generalize the implicit construction method for disjunctive cuts outlined in Algorithm 4.1. To the best of our knowledge, this is the first procedure for generating general cutting planes for non-basic solutions. The numerical results presented in Chapter 6 indicate the potential of the proposed method, since the construction times are very low, while the generated cutting planes often improve the performance significantly.

As already mentioned in the previous chapter, we focus on non-basic solutions instead of basic solutions, since we want to solve the mixed-integer quadratic program

$$
\begin{aligned}
& x \in X,\ y \in Y: \\
& \qquad \begin{array}{lrcl}
\min & \frac{1}{2}\left(x^T, y^T\right) B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix} & & \\
\text{s.t.} & A_E \begin{pmatrix} x \\ y \end{pmatrix} & = & b_E, \\
& A_I \begin{pmatrix} x \\ y \end{pmatrix} & \geq & b_I.
\end{array}
\end{aligned}
\tag{5.1}
$$

x and y denote the vectors of the continuous and integer variables, respectively, while $B \in \mathbb{R}^{n \times n}$ is a positive definite matrix and $c \in \mathbb{R}^n$ holds. X and Y are defined by the upper and lower bounds on both the continuous and the integer variables, see (1.2). n_c denotes the number of continuous variables and n_i is the number of integer variables. The total number of variables is denoted by n, i.e., $n := n_i + n_c$. Equality constraints are denoted by $A_E \in \mathbb{R}^{m_e \times n}$ and $b_E \in \mathbb{R}^{m_e}$, while inequality constraints are given by $A_I \in \mathbb{R}^{m_i \times n}$ and $b_I \in \mathbb{R}^{m_i}$. Therefore m_e denotes the number of equality constraints, while m_i is the number of inequality constraints.

To be consistent with the notation introduced in the previous chapter MIQP (5.1) is

reformulated and we obtain

$$\begin{aligned} & x \in \mathbb{R}^{n_c},\ y \in \mathbb{N}^{n_i}: \\ & \min \quad \frac{1}{2}(x^T, y^T)\, B \begin{pmatrix} x \\ y \end{pmatrix} + c^T \begin{pmatrix} x \\ y \end{pmatrix} \\ & \text{s.t.} \quad \hat{A} \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}. \end{aligned} \tag{5.2}$$

Note, that the constraints of MIQP (5.2) contain an upper and a lower bound for each continuous and integer variable. The number of constraints is again denoted by $\hat{m}$, while the number of variables n remains unchanged. We denote by $\hat{\mathbb{J}}$ the index set of all constraints of MIQP (5.2), i.e., $\hat{\mathbb{J}}$ also contains the upper and lower bounds on all variables.

In general, the solution $(\bar{x}, \bar{y}) \in X \times Y_{\mathbb{R}}$ of the continuous relaxation of MIQP (5.2) is non-basic, i.e., less than n constraints are active, see Definition 4.3. As a consequence the generation of general cutting planes is challenging, since cuts need not exist for non-basic solutions. Our task is therefore, to develop a cut generation method, that on the one hand constructs cutting planes, if they exist, while it proves their non-existence efficiently, in case no cuts exist.

We focus on disjunctive cutting planes, since these two tasks can be established also for non-basic solutions by solving the cut generation linear program (4.45). As the solution of $CGLP_k$ is computationally expensive, the generalization of Algorithm 4.1 for non-basic solutions is very attractive.

In this chapter we first briefly analyze the possibility of solving the cut generation linear program (4.45) for non-basic solutions. Since this is computationally too expensive to speed up the solution of MIQP (5.2), we propose a first generalization of the efficient cut generation procedure described by Algorithm 4.1 in Section 5.2. Since there are some cases where this generalization fails to generate the disjunctive cut, we develop an improved cut generation method in Section 5.3, that is guaranteed to construct the disjunctive cut, if it exists. Furthermore this cut generation method also efficiently proves the non-existence of disjunctive cuts. This is a very important property, since often no disjunctive cut exists for a non-basic solution $(\bar{x}, \bar{y})$ for the majority of the two-term disjunctions.

5.1 Solving the full CGLP

The reason for focusing on disjunctive cutting planes in this thesis, is the observation, that disjunctive cuts can also be constructed for non-basic solutions by solving the cut generation linear program (4.45). As already mentioned in the previous chapter, the construction of disjunctive cutting planes by solving $CGLP_k$ (4.45) is very expensive. On the one hand the dimension of $CGLP_k$ is significantly larger compared to the original MIQP (5.2). Especially, if the original mixed-integer program possesses many

constraints, a large number of variables u_i and v_i, $i = 1, \ldots, \hat{m}$ has to be introduced in the $CGLP_k$. On the other hand the cut generating linear program is mostly degenerated and cycling occurs, if no anti-cycling rules are applied. As a consequence linear solvers, e.g., CLP of Forrest et al. [54], are expected to often fail to determine the optimal solution, especially if $CGLP_k$ is set up for a non-basic solution. Furthermore, only one single cutting plane is constructed by solving the $CGLP_k$.

For mixed-integer quadratic programs no disjunctive cut exists for the majority of the fractional integer variables, see Chapter 6. In this case the optimal solution of $CGLP_k$ is already known in advance as shown in the subsequent corollary.

Corollary 5.1. *Let $(\bar{x}, \bar{y})$ be the solution of the continuous relaxation of MIQP (5.2). Let $\bar{y}_k$ be fractional and let at least one constraint $(\hat{a}_i, \hat{b}_i)$ be active at $(\bar{x}, \bar{y})$. Furthermore, assume, that no disjunctive cut exists for the disjunction*

$$y_k \leq \lfloor \bar{y}_k \rfloor \vee y_k \geq \lceil \bar{y}_k \rceil. \tag{5.3}$$

Then one optimal solution of $CGLP_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$ is given by

$$\begin{aligned} u_i &:= 0.5 \\ v_i &:= 0.5 \\ a_c &:= 0.5\hat{a}_i \\ b_c &:= 0.5\hat{b}_i, \\ u_j &:= 0, \ \forall j \in \hat{\mathbb{J}} \backslash \{i\}, \\ v_j &:= 0, \ \forall j \in \hat{\mathbb{J}} \backslash \{i\}, \\ u_0 &:= 0, \\ v_0 &:= 0, \end{aligned} \tag{5.4}$$

where $\hat{\mathbb{J}}$ denotes the index set of all constraints of MIQP (5.2).

Proof. Since by assumption no disjunctive cut exists for the disjunction (5.3), the optimal objective value of $CGLP_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$ is greater or equal to zero. Evaluating the objective function at (5.4) yields

$$\begin{aligned} a_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - b_c &= 0.5\hat{a}_i^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - 0.5\hat{b}_i \\ &= 0.5 \left(\hat{a}_i^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_i \right) \\ &= 0. \end{aligned} \tag{5.5}$$

Now we show, that (5.4) satisfies all constraints of $CGLP_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$:

Obviously the normalization constraint (4.46) is satisfied. Evaluating the remaining

constraints at (5.4) yields:

$$\begin{aligned} a_c - \hat{A}^T u + u_0 e_k &= 0.5\hat{a}_i - \hat{A}^T(0.5e_i) &&= 0 \\ a_c - \hat{A}^T v - v_0 e_k &= 0.5\hat{a}_i - \hat{A}^T(0.5e_i) &&= 0 \\ -b_c + \hat{b}^T u - u_0 \lfloor \bar{y}_k \rfloor &= -0.5\hat{b}_i + \hat{b}^T(0.5e_i) &&= 0 \\ -b_c + \hat{b}^T v + v_0 \lceil \bar{y}_k \rceil &= -0.5\hat{b}_i + \hat{b}^T(0.5e_i) &&= 0 \end{aligned} \tag{5.6}$$

Since all constraints of $\mathrm{CGLP}_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$ are satisfied and the objective value is zero, (5.4) is one of the optimal solutions of $\mathrm{CGLP}_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$, if no disjunctive cutting plane for disjunction (5.3) exists. □

Since (5.4) is the optimal solution for a vast majority of cut generating linear programs, one would like to exploit this knowledge.

Indeed there exists a variant of the simplex method called the Basis-Deficiency-Allowing Simplex developed by Pan [85], which is suited very well for this situation. The advantage of this algorithm is, that it does not work exclusively with basic solutions, in contrast to the well-known primal or dual simplex method. Instead, all calculations are based on linear independent subsets of the variables of arbitrary size. Therefore, we can directly start with the trivial, feasible solution (5.4), which is obviously non-basic with respect to the CGLP_k, and establish optimality within very few iterations. If the optimal solution of CGLP_k generates a disjunctive cutting plane, then the algorithm is expected to perform rather similar compared to the well-known primal or dual simplex method, see Pan [85].

Numerical experience shows, that the number of disjunction variables u and v, that obtain nonzero values in the optimal solution of CGLP_k is rather low, if disjunctive cutting planes exist for the optimal non-basic solution $(\bar{x}, \bar{y})$ of the continuous relaxation of MIQP (5.2). In most cases these variables corresponded to active or almost active constraints. As proved by Balas and Perregaard [17] in Theorem 4.2 the number of nonzero disjunction variables (u, v) cannot exceed n in the optimal solution of the CGLP_k, if a disjunctive cutting plane exists. Especially for non-basic solutions $(\bar{x}, \bar{y})$ the optimum of CGLP_k is often degenerated, i.e., the number of nonzero variables from (u, v) is smaller than n.

5.2 A First Efficient Cut Generation Method for Disjunctive Cuts for Non-basic Solutions

The straightforward way to construct disjunctive cutting planes by solving the full CGLP_k for every fractional integer variable is inefficient, see Section 5.1. Therefore, we want to generalize Algorithm 4.1 such that it is applicable for non-basic solutions $(\bar{x}, \bar{y}) \in X \times Y_{\mathbb{R}}$.

First we subsume some important observations from Chapter 4:

- The basic principle of Algorithm 4.1 is the equivalence of the simple disjunctive cut (4.49) and a non-trivial, feasible, basic solution of CGLP_k (4.45) established in Theorem 4.2 by Balas and Perregaard [17]. Since we cannot construct a simple disjunctive cut for a non-basic solution, Algorithm 4.1 cannot be applied in a straightforward way.
- Algorithm 4.1 can be applied without modifications also for non-basic solutions, if a cut-generating, feasible, basic solution of the CGLP_k according to Definition 4.6 is provided, since Theorem 4.3 and Theorem 4.4 do not require the solution $(\bar{x}, \bar{y})$ of the continuous relaxation to be a basic solution.
- Since the existence of disjunctive cutting planes for a non-basic solution $(\bar{x}, \bar{y})$ is not guaranteed, a cut-generating, feasible, basic solution of the CGLP_k might not exist in the general case. As a consequence, the generalization of Algorithm 4.1 needs to prove the non-existence of disjunctive cutting planes efficiently.

Now we provide an overview on the foundations of our extension of Algorithm 4.1 for non-basic solutions:

The construction of a basic solution $(\hat{x}, \hat{y})$, that is located in the disjunction induced by $\bar{y}_k$, i.e.,

$$\lfloor \bar{y}_k \rfloor \;\leq\; \hat{y}_k \;\leq\; \lceil \bar{y}_k \rceil \tag{5.7}$$

holds, is necessary to exploit the basic principle of Algorithm 4.1. Note, that the basic solution $(\hat{x}, \hat{y})$ induces a simple disjunctive cut, which is equivalent to a non-trivial, feasible, basic solution of $\mathrm{CGLP}_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$, if condition (5.7) holds. The work of Perregaard and Balas relates a basis of the original problem, i.e., the continuous relaxation of the mixed-integer problem, to a basis of the CGLP_k via the simple disjunctive cut associated with the basis and assignment (4.53), if condition (5.7) holds.

In the remainder of this section, we analyze this fundamental observation by

1. constructing a basic solution $(\hat{x}, \hat{y})$ by basis crushing.
2. constructing a basic solution $(\hat{x}, \hat{y})$ by the introduction of an artificial constraint.

5.2.1 Construction of a basic solution by basis crushing

A natural candidate for a basic solution $(\hat{x}, \hat{y}) \in X \times Y_{\mathbb{R}}$ can be determined by so-called basis crushing as follows:

Determine the largest subset of linearly independent constraints, that are active at $(\bar{x}, \bar{y})$ and denote the dimension of this subset by n_a. Then extend the subset by $n - n_a$

additional linearly independent constraints and denote the corresponding index set of these constrains by B. B forms a basis according to Definition 4.3.

Basis crushing generally deals with the question how to determine a basic solution starting at a non-basic one. Originally basis crushing is based on a QR decomposition. Beling and Megiddo [20] describe a method, that reduces the effort to $O(\hat{m}^{1.594}n)$, if fast matrix multiplications are used. Since many algorithms solving continuous quadratic programs, e.g., QL see Schittkowski [94], are based on QR decompositions, we rely on basis crushing via QR decomposition. Denoting the basic solution corresponding to B by $(\hat{x}, \hat{y})$, $(\hat{x}, \hat{y}) = \hat{A}_B^{-1}\hat{b}_B$ holds after basis crushing.

If $(\hat{x}, \hat{y})$ is constructed by basis crushing, two possibilities arise for a fractional integer variable y_k, i.e., $\bar{y}_k \notin \mathbb{Z}$. Either $\lfloor \bar{y}_k \rfloor = \lfloor \hat{y}_k \rfloor$ or $\lfloor \bar{y}_k \rfloor = \lfloor \hat{y}_k \rfloor + l$ with $l \in \mathbb{Z},\ l \neq 0$ holds. The consequences are analyzed in the subsequent corollaries.

Corollary 5.2. *Let $(\hat{x}, \hat{y})$ be a basic solution with corresponding basis B. Let $(\bar{x}, \bar{y})$ be the solution of the continuous relaxation of MIQP (5.2). Let $\lfloor \bar{y}_k \rfloor = \lfloor \hat{y}_k \rfloor$ hold.*

Then the simple disjunctive cut (4.49) induced by $(\hat{x}, \hat{y})$ for variable $\hat{y}_k$ determines a non-trivial, feasible, basic solution of CGLP$_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$ via assignment (4.53).

Proof. Due to Theorem 4.2, the simple disjunctive cut (4.49) induced by $(\hat{x}, \hat{y})$ for variable $\hat{y}_k$ is equivalent to a non-trivial, feasible, basic solution for CGLP$_k(\hat{x}, \hat{y}, \hat{A}, \hat{b})$.

Since CGLP$_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$ and CGLP$_k(\hat{x}, \hat{y}, \hat{A}, \hat{b})$ only differ in the objective function, a non-trivial, feasible, basic solution of CGLP$_k(\hat{x}, \hat{y}, \hat{A}, \hat{b})$ is also a non-trivial, feasible, basic solution of CGLP$_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$.

This proves the corollary. □

Corollary 5.3. *Let $(\hat{x}, \hat{y})$ be a basic solution with corresponding basis B. Let $(\bar{x}, \bar{y})$ be the solution of the continuous relaxation of MIQP (5.2). Let $\lfloor \bar{y}_k \rfloor = \lfloor \hat{y}_k \rfloor + l$ with $l \in \mathbb{Z},\ l \neq 0$ hold.*

Then $(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ obtained from assignment (4.53) and the simple disjunctive cut (4.49) induced by $(\hat{x}, \hat{y})$ for variable $\hat{y}_k$ is not feasible for CGLP$_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$.

Proof. Due to Theorem 4.2, the simple disjunctive cut (4.49) induced by $(\hat{x}, \hat{y})$ for variable $\hat{y}_k$ is equivalent to a non-trivial, feasible, basic solution $(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ of CGLP$_k(\hat{x}, \hat{y}, \hat{A}, \hat{b})$ with $\hat{u}_0 > 0$ and $\hat{v}_0 > 0$.

Evaluation of the constraints of CGLP$_k(\hat{x}, \hat{y}, \hat{A}, \hat{b})$ yields

$$-\hat{b}_c + \hat{b}^T\hat{u} - \hat{u}_0\lfloor \hat{y}_k \rfloor = 0, \tag{5.8}$$

which implies

$$\begin{aligned} -\hat{b}_c + \hat{b}^T\hat{u} - \hat{u}_0\lfloor \bar{y}_k \rfloor &= -\hat{b}_c + \hat{b}^T\hat{u} - \hat{u}_0(\lfloor \hat{y}_k \rfloor + l) \\ &= -\hat{u}_0 l \neq 0 \end{aligned} \tag{5.9}$$

for $\hat{u}_0 > 0$. Analogue

$$-\hat{b}_c + \hat{b}^T\hat{v} + \hat{v}_0\lceil\hat{y}_k\rceil = 0 \tag{5.10}$$

holds and therefore

$$\begin{aligned} -\hat{b}_c + \hat{b}^T\hat{v} + \hat{v}_0\lceil\bar{y}_k\rceil &= -\hat{b}_c + \hat{b}^T\hat{v} + \hat{v}_0(\lfloor\bar{y}_k\rfloor + 1) \\ &= -\hat{b}_c + \hat{b}^T\hat{v} + \hat{v}_0(\lfloor\hat{y}_k\rfloor + l + 1) \\ &= \hat{v}_0 l \neq 0 \end{aligned} \tag{5.11}$$

for $\hat{v}_0 > 0$ shows, that assignment (4.53) obtained from the simple disjunctive cut determined by $(\hat{x}, \hat{y})$ is infeasible for $\mathrm{CGLP}_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$.

Note, that the remaining constraints of both $\mathrm{CGLP}_k(\hat{x}, \hat{y}, \hat{A}, \hat{b})$ and $\mathrm{CGLP}_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$ determining the cut coefficients a_c are satisfied with equality, since they do not depend on $\lfloor\bar{y}_k\rfloor$ or $\lfloor\hat{y}_k\rfloor$.

□

As a consequence the construction of a basic solution by basis crushing, does not lead to a non-trivial, feasible, basic solution of $\mathrm{CGLP}_k(\bar{x}, \bar{y}, \hat{A}, \hat{b})$ in the general case.

5.2.2 Construction of a basic solution by the introduction of an artificial constraint

Another way for constructing a basic solution $(\hat{x}, \hat{y})$ in order to generalize Algorithm 4.1, is the introduction of an artificial constraint, denoted by

$$\hat{a}_a^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_a. \tag{5.12}$$

As we will show in the remainder of this section, the introduction of an artificial constraint has lots of advantages:

- By choosing the coefficients $\hat{a}_a$ and the right hand side $\hat{b}_a$ appropriately we can ensure, that the constructed basic solution $(\hat{x}, \hat{y})$ is located in the correct disjunction, i.e., condition (5.7) holds.
- It turns out, that the introduction of an artificial constraint (5.12) can certificate the non-existence of disjunctive cutting planes efficiently.
- The simple disjunctive cut (4.49) induced by the basic solution $(\hat{x}, \hat{y})$ is not valid for MIQP (5.2), if the corresponding basis contains an artificial constraint (5.12). The validity of the simple disjunctive cut can be recovered by a basis exchange, where the artificial constraint is replaced by one of the original constraints of MIQP (5.2).

First we show, that the introduction of an artificial constraint (5.12), yielding a basic solution $(\hat{x}, \hat{y})$ in the correct disjunction, can be used to prove the non-existence of disjunctive cutting planes for a non-basic solution $(\bar{x}, \bar{y})$ and a two-term disjunction (5.3) efficiently.

Lemma 5.1. *Let* $(\bar{x}, \bar{y})$ *be the solution of the continuous relaxation of MIQP* (5.2). *Furthermore, let*

$$\hat{A}^1 \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}^1 \tag{5.13}$$

with $\hat{A}^1 \in \mathbb{R}^{\hat{m}^1 \times n}$, $\hat{b}^1 \in \mathbb{R}^{\hat{m}^1}$ *and*

$$\hat{A}^2 \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}^2 \tag{5.14}$$

with

$$\hat{A}^2 := \begin{bmatrix} \hat{A}^1 \\ \tilde{A}^2 \end{bmatrix}, \tag{5.15}$$

$$\hat{b}^2 := \begin{bmatrix} \hat{b}^1 \\ \tilde{b}^2 \end{bmatrix} \tag{5.16}$$

and $\tilde{A}^2 \in \mathbb{R}^{\tilde{m}^2 \times n}$, $\tilde{b}^2 \in \mathbb{R}^{\tilde{m}^2}$, $\hat{A}^2 \in \mathbb{R}^{\hat{m}^2 \times n}$, $\hat{b}^2 \in \mathbb{R}^{\hat{m}^2}$ *be the description of two polyhedra* P^1 *and* P^2, *such that* P^1 *is a relaxation of* P^2, *i.e.,* $P^2 \subset P^1$. *Let* $(\hat{x}, \hat{y})$ *be a basic solution of* $\hat{A}^1$ *with index set* B *and*

$$\lfloor \bar{y}_k \rfloor \leq \hat{y}_k \leq \lceil \bar{y}_k \rceil. \tag{5.17}$$

Let $(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ *be the non-trivial basic solution of* $CGLP_k(\bar{x}, \bar{y}, \hat{A}^1, \hat{b}^1)$ *induced by* B. *Assume, that* $(z^*_{CGLP_k})^1 := \hat{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_c$ *is the optimal objective value of* $CGLP_k(\bar{x}, \bar{y}, \hat{A}^1, \hat{b}^1)$.
Then the optimal value of the $CGLP_k(\bar{x}, \bar{y}, \hat{A}^2, \hat{b}^2)$ *is less or equal to* $(z^*_{CGLP_k})^1$.

Proof. By assumption, $(z^*_{\mathrm{CGLP}_k})^1 := \hat{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_c$ is the optimal objective value of $\mathrm{CGLP}_k(\bar{x}, \bar{y}, \hat{A}^1, \hat{b}^1)$. This cut is equivalent to the simple disjunctive cut (4.49) induced by B according to Theorem 4.2. Due to the construction of $\hat{A}^2$ and $\hat{b}^2$, B is also a basis of $\hat{A}^2$. As a consequence of Theorem 4.2 the simple disjunctive cut induced by B is also equivalent to a non-trivial, feasible, basic solution of $\mathrm{CGLP}_k(\bar{x}, \bar{y}, \hat{A}^2, \hat{b}^2)$. Therefore, $(z^*_{\mathrm{CGLP}_k})^1 \geq (z^*_{\mathrm{CGLP}_k})^2$ holds, which proves the Lemma. □

The following corollary is a direct consequence of Lemma 5.1 and can be applied to prove the non-existence of disjunctive cutting planes for non-basic solutions $(\bar{x}, \bar{y})$, if an artificial constraint (5.12) has been introduced.

Corollary 5.4. *Let* $(\bar{x}, \bar{y})$ *be the solution of the continuous relaxation of MIQP* (5.2). *Furthermore, let*

$$\hat{A}^1 \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}^1 \tag{5.18}$$

with $\hat{A}^1 \in \mathbb{R}^{\hat{m}^1 \times n}$, $\hat{b}^1 \in \mathbb{R}^{\hat{m}^1}$ *and*

$$\hat{A}^2 \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}^2 \tag{5.19}$$

with

$$\hat{A}^2 := \begin{bmatrix} \hat{A}^1 \\ \tilde{A}^2 \end{bmatrix}, \tag{5.20}$$

$$\hat{b}^2 := \begin{bmatrix} \hat{b}^1 \\ \tilde{b}^2 \end{bmatrix} \tag{5.21}$$

and $\tilde{A}^2 \in \mathbb{R}^{\tilde{m}^2 \times n}$, $\tilde{b}^2 \in \mathbb{R}^{\tilde{m}^2}$, $\hat{A}^2 \in \mathbb{R}^{\hat{m}^2 \times n}$, $\hat{b}^2 \in \mathbb{R}^{\hat{m}^2}$ *be the description of two polyhedra* P^1 *and* P^2, *such that* P^1 *is a relaxation of* P^2, *i.e.,* $P^2 \subset P^1$. *Let* $(\hat{x}, \hat{y})$ *be a basic solution of* $\hat{A}^2$ *with index set* B *and*

$$\lfloor \bar{y}_k \rfloor \leq \hat{y}_k \leq \lceil \bar{y}_k \rceil. \tag{5.22}$$

Let $(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ *be the non-trivial basic solution of* $CGLP_k(\bar{x}, \bar{y}, \hat{A}^2, \hat{b}^2)$ *induced by* B. *Assume, that* $(z^*_{CGLP_k})^2 := \hat{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_c \geq 0$ *is the optimal objective value of* $CGLP_k(\bar{x}, \bar{y}, \hat{A}^2, \hat{b}^2)$.

Then no disjunctive cutting planes for the disjunction (5.3) *determined by* $\bar{y}_k$ *exist for polyhedron* P^1, *i.e., the optimal objective value of* $CGLP_k(\bar{x}, \bar{y}, \hat{A}^1, \hat{b}^1)$ *is greater or equal zero.*

Proof. Applying Lemma 5.1 yields $(z^*_{\mathrm{CGLP}_k})^1 \geq (z^*_{\mathrm{CGLP}_k})^2 \geq 0$. Note, that $(z^*_{\mathrm{CGLP}_k})^1 \geq 0$ proves the non-existence of disjunctive cutting planes for the corresponding disjunction (5.3). □

In the remainder of this subsection, different artificial constraints are analyzed with respect to their suitability for a generalization of Algorithm 4.1. As mentioned at the beginning of this subsection any artificial constraint has to ensure, that the constructed basic solution $(\hat{x}, \hat{y})$ is located in the considered disjunction, i.e., (5.7) holds. Furthermore, Corollary 5.4, which might give a certificate for the non-existence of disjunctive cutting planes for the disjunction (5.3), can be applied independently of the choice of the artificial constraint.

As a consequence, the artificial constraint has to be determined, such that it can be removed from the basis easily to obtain a valid inequality after a basis exchange.

As a first attempt we choose the artificial constraint (5.12) to be either

$$y_k = \lfloor \bar{y}_k \rfloor \tag{5.23}$$

or

$$y_k = \lceil \bar{y}_k \rceil. \tag{5.24}$$

Then the corresponding basic solution $(\hat{x}, \hat{y})$ is located on the boundary of the disjunction determined by $\bar{y}_k$, where $(\bar{x}, \bar{y})$ is the solution of the continuous relaxation of MIQP (5.2) to be cut off, i.e., $\hat{y}_k = \lfloor \bar{y}_k \rfloor$ or $\hat{y}_k = \lceil \bar{y}_k \rceil$ holds.

Note, that we can determine $(\hat{x}, \hat{y})$ trivially by including one upper or lower bound for each continuous and integer variable apart from variable y_k in the corresponding basis B and choosing the artificial constraint to be (5.23) or (5.24).

Instead of constructing the basis with upper or lower bounds on the variables, we can also form the basis with (5.23) or (5.24) and those constraints that are linearly independent and possess the lowest slack values $\bar{s}_j$ at $(\bar{x}, \bar{y})$, with

$$\bar{s}_j := \hat{a}_j \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j, \quad j \in \hat{\mathbb{J}}. \tag{5.25}$$

It turns out, that choosing the artificial constraint to be either (5.23) or (5.24) has a disadvantage:

If either (5.23) or (5.24) are included in the basis forming $(\hat{x}, \hat{y})$, the k-th row of the inverse of the basis matrix is given by either e_k or $-e_k$, i.e.,

$$(\hat{A}_B^{-1})_k = e_k \tag{5.26}$$

or

$$(\hat{A}_B^{-1})_k = -e_k \tag{5.27}$$

holds.

Since the reduced costs in Algorithm 4.1 are determined by this row, see Theorem 4.3, we have to perform perturbation Step 7 of Algorithm 4.1 in order to continue. From a computational point of view, the perturbation step should be avoided, if possible, see Perregaard [86].

5.2.3 A suitable artificial Constraint for an efficient Cut-Generation Method for non-basic Solutions

We want to choose the artificial constraint (5.12) such that

- the constructed basic solution $(\hat{x}, \hat{y})$ is located in the correct disjunction,
- we can prove the non-existence of disjunctive cutting planes efficiently,
- we can remove the artificial constraint from the basis easily.

Corollary 5.4 is to be exploited, to prove the non-existence of disjunctive cutting planes for a given disjunction. This means, that we want to construct a non-trivial, feasible, basic solution of the extended $\text{CGLP}_k\left(\bar{x}, \bar{y}, \left[\hat{A}^T, \hat{a}_a\right]^T, \left[\hat{b}^T, \hat{b}_a\right]^T\right)$, such that the corresponding simple disjunctive cutting plane does not cut off the relaxed solution $(\bar{x}, \bar{y})$ of MIQP (5.2).

Recalling Lemma 4.1 and Definition 4.6 we know that a trivial, feasible solution $(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ of any CGLP_k yields no cutting plane, since $(\hat{a}_c, \hat{b}_c)$ corresponds to a non-negative linear combination of the constraints associated with the variables $\hat{u}_i > 0$ and $\hat{v}_i > 0$. This means that the corresponding linear inequality given by $(\hat{a}_c, \hat{b}_c)$ is not violated by any feasible point (x, y) satisfying $\hat{A}\begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}$. As a consequence, it fits to Corollary 5.4, but in order to extend Algorithm 4.1 we need to work with non-trivial, feasible, basic solution, e.g., for calculating reduced costs (4.55).

The subsequent corollary shows how both requirements can be accomplished.

Corollary 5.5. *Let* $(\hat{x}, \hat{y})$ *be a basic solution satisfying*

$$\hat{y}_k = \lceil \bar{y}_k \rceil - \varepsilon, \tag{5.28}$$

where $(\bar{x}, \bar{y})$ *is the solution of the continuous relaxation of MIQP* (5.2) *and* $\varepsilon \in [0, 1]$ *is a constant. Denote the basis forming* $(\hat{x}, \hat{y})$ *by* B *and let* B *contain an artificial constraint* $(\hat{a}_a, \hat{b}_a(\varepsilon))$*, where the right hand side also depends on* ε *and is chosen such that* (5.28) *holds.*

For $\varepsilon = 0$ *and* $\varepsilon = 1$*, the simple disjunctive cut* (4.49) *induced by* $(\hat{x}, \hat{y})$ *determines a non-trivial, feasible, basic solution of* $CGLP_k\left(\bar{x}, \bar{y}, \left[\hat{A}^T, \hat{a}_a\right]^T, \left[\hat{b}^T, \hat{b}_a(\varepsilon)\right]^T\right)$ *via assignment* (4.53)*, that does not cut off* $(\bar{x}, \bar{y})$*.*

Proof. The simple disjunctive cut induced by $(\hat{x}, \hat{y})$ is given by

$$\begin{aligned}
\pi_0 &= (\hat{y}_k - \lfloor \bar{y}_k \rfloor)(\lceil \bar{y}_k \rceil - \hat{y}_k) = (1 - \varepsilon)\varepsilon, \\
\pi_j^1 &= (\lceil \bar{y}_k \rceil - \hat{y}_k)\left(\begin{bmatrix} \hat{A} \\ \hat{a}_a \end{bmatrix}_B^{-1}\right)_{kj} = -\varepsilon\left(\begin{bmatrix} \hat{A} \\ \hat{a}_a \end{bmatrix}_B^{-1}\right)_{kj}, \\
\pi_j^2 &= (\hat{y}_k - \lfloor \bar{y}_k \rfloor)\left(\begin{bmatrix} \hat{A} \\ \hat{a}_a \end{bmatrix}_B^{-1}\right)_{kj} = (1 - \varepsilon)\left(\begin{bmatrix} \hat{A} \\ \hat{a}_a \end{bmatrix}_B^{-1}\right)_{kj}, \\
\pi_j &= \max\{\pi_j^1, \pi_j^2\} \quad \forall j \in B,
\end{aligned} \tag{5.29}$$

Since $\hat{y}_k$ satisfies (5.28) with $\varepsilon \in [0, 1]$, $(\hat{x}, \hat{y})$ is located in the correct disjunction, i.e., (5.7) holds. As a consequence we obtain a non-trivial, feasible, basic solution

$(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ for $\mathrm{CGLP}_k\left(\bar{x}, \bar{y}, \left[\hat{A}^T, \hat{a}_a\right]^T, \left[\hat{b}^T, \hat{b}_a(\varepsilon)\right]^T\right)$ by assignment (4.53) with (5.29). For $\varepsilon = 0$ this non-trivial, feasible, basic solution satisfies $\hat{u}_0 = 0$, while $\hat{v}_0 = 0$ holds for $\varepsilon = 1$, see assignment (4.53) stating

$$\hat{u}_0 := (\lceil \bar{y}_k \rceil - \hat{y}_k)\,\theta \tag{5.30}$$

$$\hat{v}_0 := (\hat{y}_k - \lfloor \bar{y}_k \rfloor)\,\theta, \tag{5.31}$$

with $\theta > 0$.

Due to Lemma 4.1, the inequality defined by $(\hat{a}_c, \hat{b}_c)$ does not cut off $(\bar{x}, \bar{y})$, which proves the corollary. □

Figure 5.1 illustrates the previous corollary. The basic solution $(\hat{x}, \hat{y})$ is determined by the constraints $\hat{a}_l$ and $\hat{a}_j$ and the corresponding simple disjunctive cut is c. Moving constraint $\hat{a}_j$ by parallel translation yields constraint $\hat{a}'_j$ determining basic solution $(\hat{x}', \hat{y}')$ and the simple disjunctive cut c'. The cut c' is weaker than c. If we repeat this procedure we obtain basic solution $(\hat{x}'', \hat{y}'')$ determined by constraint $\hat{a}''_j$ and $\hat{a}_l$ with $\hat{y}'' = \lceil \hat{y} \rceil$. As a consequence the associated simple disjunctive cut is equivalent to constraint $\hat{a}_l$, which implies, that the feasible region is not truncated.

Note, that it is possible to apply the steps of Algorithm 4.1 to the non-trivial, feasible, basic solution of the CGLP_k constructed by Corollary 5.5 with $\varepsilon = 0$ or $\varepsilon = 1$. The reason is, that although $(\hat{x}, \hat{y})$ is located on the boundary of the disjunction, i.e., (5.23) or (5.24) hold, $(\hat{x}, \hat{y})$ induces a non-trivial, feasible, basic solution of the CGLP_k. The reduced cost for u_i and v_i with $i \notin B$ can be evaluated for this non-trivial, feasible, basic solution of the CGLP_k, since all terms in formulas (4.55) depend on the corresponding basis matrix $\begin{bmatrix} \hat{A} \\ \hat{a}_a \end{bmatrix}_B$ and its inverse. The sets M_1 and M_2 are defined according to the corresponding simple disjunctive cut (4.49) by the sign of the corresponding element of $\begin{bmatrix} \hat{A} \\ \hat{a}_a \end{bmatrix}_B^{-1}$, i.e.,

$$M_1 := \{j \in B : \bar{a}_{kj} < 0 \vee (\bar{a}_{kj} = 0 \wedge \bar{a}_{ij} > 0)\}$$

and

$$M_2 := B \backslash M_1$$

for determining r_{u_i} and

$$M_1 := \{j \in B : \bar{a}_{kj} < 0 \vee (\bar{a}_{kj} = 0 \wedge \bar{a}_{ij} < 0)\}$$

and

$$M_2 := B \backslash M_1$$

for determining r_{v_i}. Note, that we applied notation (4.25). Therefore it is possible to run Algorithm 4.1 starting with this non-trivial, feasible, basic solution of the $CGLP_k$.

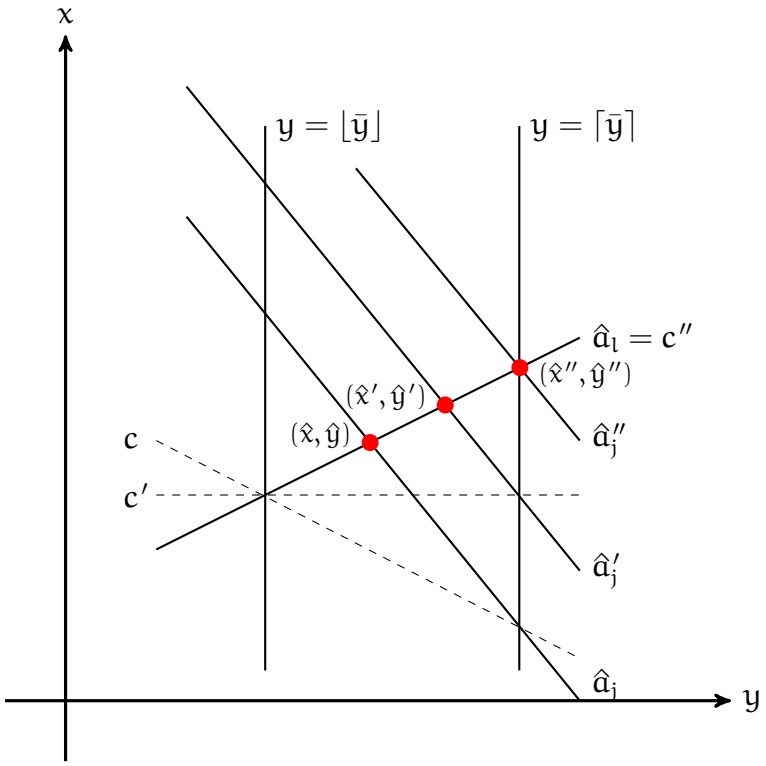

Fig. 5.1: Simple Disjunctive Cut in the Limit

As a consequence, the non-trivial, feasible, basic solution of the $CGLP_k$ obtained from the simple disjunctive cut (5.29) due to assignment (4.53) ensures that the solution $(\bar{x}, \bar{y})$ of the continuous relaxation of MIQP (5.2) is not cut off. This allows the application of Corollary 5.4, which might give a certificate for the non-existence of disjunctive cutting planes for the current disjunction.

Nevertheless, we still have to deal with the situation, that Corollary 5.4 cannot be applied, since some reduced costs (4.55) are negative. In this situation the artificial constraint (5.12) needs to be removed, in order to construct a valid cutting plane or at least a valid inequality.

In the sequel we propose an the construction method for an artificial constraint, which ensures, that the artificial constraint can be removed from the basis by a basis exchange under certain conditions. Furthermore, the constructed artificial constraint ensures that the basic solution $(\hat{x}, \hat{y})$ is located on the boundary of the disjunction, i.e., (5.23) or (5.24) hold. As a consequence the solution $(\bar{x}, \bar{y})$ of the continuous relaxation of MIQP (5.2) is not cut off by the induced simple disjunctive cut, see Corollary 5.5.

Algorithm 5.1. *Let* $(\bar{x}, \bar{y})$ *be the non-basic solution of the continuous relaxation of MIQP* (5.2). *Let* y_k *be an integer variable, that is fractional at* $(\bar{x}, \bar{y})$, *i.e.,* $\bar{y}_k \notin \mathbb{N}$.

1. *Determine a not necessarily feasible, basic solution* $(\hat{x}, \hat{y})$ *with the corresponding basis* $\tilde{B}$ *and basic matrix* $\hat{A}_{\tilde{B}}$*, such that* $\tilde{B}$ *contains either constraint*

$$y_k = \lfloor \bar{y}_k \rfloor \tag{5.32}$$

or

$$\hat{y}_k = \lceil \bar{y}_k \rceil, \tag{5.33}$$

where we denote the corresponding index within basis $\tilde{B}$ *by* $\bar{k}$*. I.e., either* $\hat{y}_k = \lfloor \bar{y}_k \rfloor$*, or alternatively* $\hat{y}_k = \lceil \bar{y}_k \rceil$ *holds.*

2. *Determine the artificial constraint* $(\hat{a}_a, \hat{b}_a)$ *by*

$$\begin{aligned} \hat{a}_a^T &:= -e^T \hat{A}_{\tilde{B}}, \\ \hat{b}_a &:= -e^T \hat{A}_{\tilde{B}} \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}, \end{aligned} \tag{5.34}$$

where $e \in \mathbb{R}^n$ *is the vector of all ones.*

3. *Replace constraint* $y_k = \lfloor \bar{y}_k \rfloor$*, or alternatively* $y_k = \lceil \bar{y}_k \rceil$*, by the artificial constraint* $(\hat{a}_a, \hat{b}_a)$ *and denote the index set by* B*, i.e.*

$$B := \tilde{B} \backslash \{\bar{k}\} \cup \{a\}, \tag{5.35}$$

where a *is the index of the artificial constraint* (5.34)*.*

4. ***RETURN*** *artificial constraint* (5.34) *and basis* B *with the corresponding basic solution* $(\hat{x}, \hat{y})$*.*

Note, that the basic solution $(\hat{x}, \hat{y})$ generated in Step 1 can always be determined by one bound on each variable apart from y_k and the constraint

$$y_k = \lfloor \bar{y}_k \rfloor. \tag{5.36}$$

or

$$y_k = \lceil \bar{y}_k \rceil. \tag{5.37}$$

Alternatively, basis crushing starting at $(\bar{x}, \bar{y})$ and with the constraint (5.36) or (5.37) can be applied to obtain $(\hat{x}, \hat{y})$, such that the basis $\tilde{B}$ contains as a subset all linear independent constraints, that are active at $(\bar{x}, \bar{y})$.

The subsequent corollary motivates the construction of the artificial constraint by assignment (5.34) in Algorithm 5.1.

Corollary 5.6. *Let the artificial constraint* $(\hat{a}_a, \hat{b}_a)$ *and the basis* B *with corresponding basic solution* $(\hat{x}, \hat{y})$ *be constructed by Algorithm 5.1. Furthermore let* $\tilde{B}$ *be the basis constructed in Step 1 of Algorithm 5.1.*

Then $(\hat{x}, \hat{y})$ *is the unique optimal solution of the linear program*

$$\begin{aligned} x \in \mathbb{R}^{n_c}, y \in \mathbb{R}^{n_i} : \\ \min \quad & \tilde{a}_a^T \begin{pmatrix} x \\ y \end{pmatrix} \\ \textit{s.t.} \quad & \hat{A}_{\tilde{B}} \begin{pmatrix} x \\ y \end{pmatrix} = \hat{b}_{\tilde{B}}, \end{aligned} \tag{5.38}$$

with $\tilde{a}_a := -\hat{a}_a$.

Proof. Note, that the negated coefficients of the artificial constraint determined by Algorithm 5.1, correspond to the vector of the objective function in LP (5.38).

The reduced costs with respect to a general objective function $c^T \begin{pmatrix} x \\ y \end{pmatrix}$ are given by $c_{\tilde{B}^\perp}^T - c_{\tilde{B}}^T \hat{A}_{\tilde{B}}^{-1} \hat{A}_{\tilde{B}^\perp}$, where $\tilde{B}$ denotes the indices of the basic variables, while $\tilde{B}^\perp$ denotes the indices of the non-basic variables, see e.g., Jarre and Stoer [66]. For the linear program (5.38) $\hat{A}_{\tilde{B}^\perp} = -I$ holds and we can set $c_{\tilde{B}^\perp} := 0$. Therefore the optimality conditions for $(\hat{x}, \hat{y})$ and LP (5.38) are

$$-c_{\tilde{B}}^T \hat{A}_{\tilde{B}}^{-1}(-I) > 0, \tag{5.39}$$

$$c_{\tilde{B}}^T \hat{A}_{\tilde{B}}^{-1} > 0. \tag{5.40}$$

The negated coefficients $\tilde{a}_a$ satisfy the optimality conditions due to assignment (5.34), which proves the corollary. □

Since $(\hat{x}, \hat{y})$ is the optimal solution of LP (5.38), $(\hat{x}, \hat{y})$ is the only point satisfying all constraints $\hat{A}_{\tilde{B}} \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_{\tilde{B}}$ and $\tilde{a}_a^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \tilde{b}_a$, with $\tilde{b}_a := -\hat{b}_a$. Therefore, the artificial constraint $(\hat{a}_a, \hat{b}_a)$ determined by Algorithm 5.1 with $\hat{a}_a = -\tilde{a}_a$ and $\hat{b}_a = -\tilde{b}_a$ is dominated by the constraints included in $\tilde{B}$, see also Figure 5.2, i.e., $(\hat{x}, \hat{y})$ is the only point, where the artificial constraint obtained by (5.34) is active and all constraints forming $\tilde{B}$ are satisfied.

Under certain conditions the proposed construction of the artificial constraint can be exploited, in order to remove the artificial constraint determined by (5.34) to obtain a valid inequality. This is proved in the subsequent lemma.

Lemma 5.2. *Let the artificial constraint* $(\hat{a}_a, \hat{b}_a)$ *and the basis* B *with corresponding basic solution* $(\hat{x}, \hat{y})$ *be constructed by Algorithm 5.1. Let the artificial constraint be indexed by* a*, while all other constraints are indexed by* $1, \ldots, n-1$*, i.e.,* $B =$

$\{1, \ldots, n-1, a\}$. *Denote the objective value of the induced non-trivial, feasible, basic solution of the CGLP$_k$ by* $\hat{z}_{\mathrm{CGLP}_k}$.

Let $B_1 := B\backslash\{l\} \cup \{j\}$ *be a neighboring basis with* $l \in B$ *and* $j \notin B$*, that induces a non-trivial, feasible, basic solution of CGLP$_k$, where the corresponding objective value is denoted by* $(z_{\mathrm{CGLP}_k})^1$*, with* $(z_{\mathrm{CGLP}_k})^1 < \hat{z}_{\mathrm{CGLP}_k}$.

Then the basis determined by the constraints

$$\hat{a}_i^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_i, \quad \forall i \in B\backslash\{a\} \tag{5.41}$$

and

$$\hat{a}_j^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_j, \tag{5.42}$$

induces a non-trivial, feasible, basic solution of the CGLP$_k$, where the corresponding objective value is denoted by z_{CGLP_k}*, with* $z_{\mathrm{CGLP}_k} < \hat{z}_{\mathrm{CGLP}_k}$.

Proof. By assumption basis B_1 induces a non-trivial, feasible, basic solution of CGLP_k with $(z_{\mathrm{CGLP}_k})^1 < \hat{z}_{\mathrm{CGLP}_k}$. By construction B_1 is adjacent to B, which ensures, that the subsequent conditions hold for the corresponding basic solution $(\hat{x}_1, \hat{y}_1)$.

$$\begin{aligned} \lfloor \bar{y}_k \rfloor \leq (\hat{y}_1)_k \leq \lceil \bar{y}_k \rceil, \\ \hat{a}_i^T \begin{pmatrix} \hat{x}_1 \\ \hat{y}_1 \end{pmatrix} &= \hat{b}_i, \ \forall i \in B\backslash\{l, a\}, \\ \hat{a}_j^T \begin{pmatrix} \hat{x}_1 \\ \hat{y}_1 \end{pmatrix} &= \hat{b}_j, \\ \hat{a}_a^T \begin{pmatrix} \hat{x}_1 \\ \hat{y}_1 \end{pmatrix} &= \hat{b}_a. \end{aligned} \tag{5.43}$$

$(\hat{x}_1, \hat{y}_1)$ is constructed by replacing some constraint l

$$\hat{a}_l^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_l, \quad l \in B \tag{5.44}$$

with constraint j

$$\hat{a}_j^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_j, \quad j \notin B. \tag{5.45}$$

Without loss of generality, we consider the two-dimensional case with $n = 2$, since it is sufficient to work in the null-space of the matrix formed by the constraints

$$\hat{a}_i^T \begin{pmatrix} x \\ y \end{pmatrix} = \hat{b}_i, \quad \forall i \in B\backslash\{l, a\}. \tag{5.46}$$

Then the basic solution $(\hat{x}, \hat{y})$ determined by

$$\begin{aligned} \hat{a}_a^T \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} &= \hat{b}_a, \\ \hat{a}_l^T \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} &= \hat{b}_l. \end{aligned} \tag{5.47}$$

induces a non-trivial, feasible, basic solution $(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ of CGLP_k with objective value $\hat{z}_{\mathrm{CGLPK}}$.

By construction the intersection of the artificial constraint

$$\hat{a}_a^T \begin{pmatrix} x \\ y \end{pmatrix} = \hat{b}_a \tag{5.48}$$

and the polyhedron P given by

$$P := \left\{ (x, y) \in \mathbb{R}^2 : \hat{a}_l^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_l, \quad y \geq \lfloor \bar{y} \rfloor, \quad y \leq \lceil \bar{y} \rceil \right\} \tag{5.49}$$

is the basic solution $(\hat{x}, \hat{y})$ due to Corollary 5.6, i.e.,

$$P \cap \left\{ (x, y) \in \mathbb{R}^2 : \hat{a}_a^T \begin{pmatrix} x \\ y \end{pmatrix} = \hat{b}_a \right\} = \{(\hat{x}, \hat{y})\}. \tag{5.50}$$

As by construction $(\hat{x}, \hat{y})$ is located on the boundary of the disjunction, i.e., (5.23) or (5.24) hold, $(\hat{a}_c, \hat{b}_c)$ is equivalent to the coefficients $(\hat{a}_l, \hat{b}_l)$ of constraint

$$\hat{a}_l^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_l, \tag{5.51}$$

see also Lemma 4.1 and assignment (4.53).

By assumption the exchange of constraint l by constraint j yields the basis B_1 which induces a non-trivial, feasible, basic solution of the CGLP_k with $(z_{\mathrm{CGLP}_k})^1 < \hat{z}_{\mathrm{CGLP}_k}$. Therefore the intersection of the constraint

$$\hat{a}_j^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_j, \tag{5.52}$$

with the polyhedron P given by (5.50) needs to truncate the polyhedron P, i.e.,

$$P \cap \left\{ (x, y) \in \mathbb{R}^2 : \hat{a}_j^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_j \right\} \subsetneq P. \tag{5.53}$$

As a consequence there exists a basis determined by the intersection of constraint l and j within P. This basis induces a non-trivial, feasible, basic solution of the CGLP_k, with objective value z_{CGLP_k} satisfying $z_{\mathrm{CGLP}_k} < \hat{z}_{\mathrm{CGLP}_k}$ due to the construction of the artificial constraint, see Corollary 5.6.

This proves the lemma. □

Figure 5.2 illustrates the situation, where $(\hat{x}_2, \hat{y}_2)$ denotes the alternative basic solution determined by the intersection of constraint l and j and c_1 denotes the improving simple disjunctive cut induced by both $(\hat{x}_1, \hat{y}_1)$ and $(\hat{x}_2, \hat{y}_2)$.

Note, that the cut induced by $(\hat{x}_1, \hat{y}_1)$ might differ from the one generated by $(\hat{x}_2, \hat{y}_2)$, in case that constraint j intersects the hyperplane $y = \lfloor \bar{y} \rfloor$ within the polyhedron P.

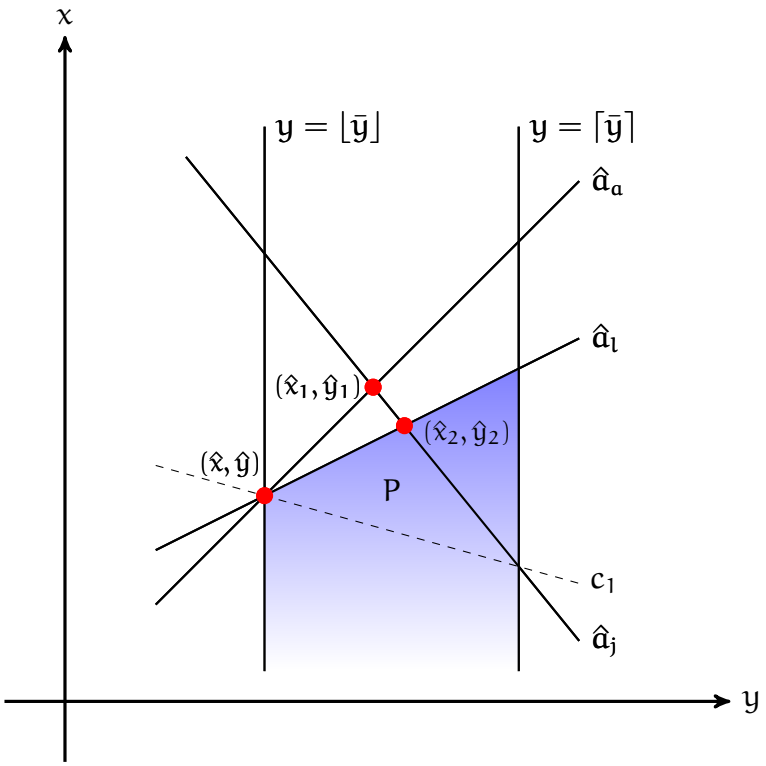

Fig. 5.2: Alternative Simple Disjunctive Cut

Based on the previous considerations we propose the following extension of Algorithm 4.1. It can efficiently generate disjunctive cutting planes and in some cases the non-existence of disjunctive cutting planes for the current disjunction (4.27) can be proved.

Algorithm 5.2. *1. Solve the continuous relaxation of MIQP* (5.2)*. Let* $(\bar{x}, \bar{y})$ *be the optimal solution with* $\lfloor \bar{y}_k \rfloor < \bar{y}_k < \lceil \bar{y}_k \rceil$*, i.e.,* $\bar{y}_k$ *is fractional.*

2. Determine a basic solution, if necessary:

If $(\bar{x}, \bar{y})$ *is no basic solution,* **then** *determine an artificial constraint* $\hat{a}_a^T \begin{pmatrix} x \\ y \end{pmatrix} \geq \hat{b}_a$ *and a basis* B *with corresponding basic solution* $(\hat{x}, \hat{y})$ *by Algorithm 5.1.*

Else *denote the basis determining* $(\bar{x}, \bar{y})$ *by* B*.*

Determine the non-trivial, feasible, basic solution $(\hat{a}_c, \hat{b}_c, \hat{u}, \hat{u}_0, \hat{v}, \hat{v}_0)$ *of the CGLP*$_k$ *induced by basis* B.

3. *Compute the reduced costs* (4.55) *according to Step 2 of Algorithm 4.1.*

4. *Remove artificial constraint, if necessary:*

   If *an artificial constraint with index* a *is included in* B, **then** *set the index* j_* *of the basic constraint to be removed from* B *to* a, *i.e., set* $j_* = a$. *Set the index* i^* *of the non-basic constraint to be included in the basis to the index* i *of that non-basic constraint* $(\hat{a}_i, \hat{b}_i)$, $i \notin B$ *that possesses the lowest negative reduced costs for either* $\hat{u}_i$ *or* $\hat{v}_i$ *and yields a basic solution within the disjunction induced by* $\bar{y}_k$ *after the basis exchange* i^* *for* j_*, *i.e., condition* (4.89) *is satisfied for* $\gamma := -\frac{\bar{a}_{kj_*}}{\bar{a}_{i^*j_*}}$.

   If i^* *exists,* **then GOTO** *Step 7.*

   Else *,i.e., there exists no neighboring basic solution within the disjunction induced by* $\bar{y}_k$, *that improves the current cut* $(\bar{a}_c, \bar{b}_c)$, **STOP**.

5. *Determine constraint* i^* *to be included in the basis according to Step 3 of Algorithm 4.1.*

6. *Determine constraint* j_* *to be removed from the basis according to Step 4 of Algorithm 4.1.*

7. *Execute basis exchange according to Step 5 of Algorithm 4.1.*

8. **GOTO** *Step 3.*

9. *Execute perturbation according to Step 7 of Algorithm 4.1.*

There is one situation, where Algorithm 5.2 is not able to construct a disjunctive cut:

As long as the current basis does not induce a cut-generating, feasible, basic solution of the CGLP$_k$, no neighboring basis inducing a non-trivial, feasible, basic solution of the CGLP$_k$ with reduced objective value might exist, even if a disjunctive cut exists for the current disjunction. Then, Lemma 5.2 is not applicable and the artificial constraint cannot be removed.

This situation might occur, since a non-trivial, feasible, basic solution of the CGLP$_k$, which is not a cut-generating, feasible, basic solution of the CGLP$_k$ can be improved by a trivial, feasible solution of the CGLP$_k$, e.g., corresponding to an active constraint (5.4). The identification of this situation is straightforward, since there exist non-basic constraints $(\hat{a}_i, \hat{b}_i)$ $i \notin B$ possessing negative reduced costs for the corresponding $\hat{u}_i$ or $\hat{v}_i$, but condition (4.89) with $\gamma := -\frac{\bar{a}_{kj}}{\bar{a}_{ij}}$ is not satisfied for $j = a$, where a is the index of the artificial constraint in B.

Nevertheless we can reduce the probability, that such a situation occurs, if we take the slack value $\bar{s}_j,\ \forall j \in \hat{\mathbb{J}}$ (5.25) during basis crushing into account, i.e., while determining the basic solution $(\hat{x}, \hat{y})$ in Step 1 of Algorithm 5.1. Starting with all linear independent active constraints we can successively include the constraint $(\hat{a}_j, \hat{b}_j),\ j \notin B$ with the lowest slack value $\bar{s}_j$ in B, if it is linear independent subject to all constraints already included in B.

If Algorithm 5.2 is not successful, i.e., it cannot construct a disjunctive cut, although one exists, then either all adjacent basic solutions are located outside the current disjunction, i.e., $\tilde{y}_k$ does not satisfy

$$\lfloor \bar{y}_k \rfloor \leq \tilde{y}_k \leq \lceil \bar{y}_k \rceil, \tag{5.54}$$

for any adjacent basic solution $(\tilde{x}, \tilde{y})$. The other possibility is that the non-trivial, feasible, basic solution of the $CGLP_k$ induced by any basis, that is adjacent to the current basis, possesses a higher objective value than the non-trivial, feasible, basic solution of the $CGLP_k$ induced by the current basis, see also Lemma 5.2.

5.3 An Improved Cut Generation Method for Disjunctive Cuts for Non-basic Solutions

In the previous section, we proposed an efficient cut generation procedure for disjunctive cutting planes, that can be applied for non-basic solutions. In principle it provides most of the desired features, e.g., it detects the non-existence of disjunctive cuts efficiently. But there might be situations, in which the proposed algorithm fails to construct the disjunctive cut. To overcome this problem we propose an improved cut generation method, which exploits some of the observations presented so far.

In the remainder of this chapter, we are only interested in constructing cutting planes, that possess a minimum strength $\varepsilon > 0$, i.e.,

$$\hat{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_c \ \leq \ -\varepsilon. \tag{5.55}$$

holds for the cutting plane determined by $(\hat{a}_c, \hat{b}_c)$. Weaker cutting planes are neglected, e.g., for numerical reasons.

The cut generation method proposed in this section again relies on the introduction of an artificial constraint (5.12). In contrast to the previously described cut generation method, the right hand side of the artificial constraint is adapted in each iteration until it is removed by the basis exchange procedure of the original Algorithm 4.1. In each iteration the current cut possesses strength $\varepsilon > 0$, i.e., (5.55) holds. This allows the application of Lemma 5.4 to efficiently prove the non-existence of disjunctive cutting planes stronger than ε.

We start by constructing a basic solution $(\hat{x}, \hat{y})$ according to the subsequent Algorithm 5.3.

Algorithm 5.3. *Let* $(\bar{x}, \bar{y})$ *be the optimal non-basic solution of the continuous relaxation of MIQP* (5.2)*. Let* $n_a < n$ *linearly independent constraints be active at* $(\bar{x}, \bar{y})$ *and denote the corresponding index set by* B*. Let* e_k *be linearly independent w.r.t. the constraints contained in* B*.*

1. *Add constraint*
$$y_k = \bar{y}_k \tag{5.56}$$
to the index set B*. Denote the corresponding index by* $\bar{k}$*.*

2. *Continue until* $|B| = n$*:*

   (a) *Include constraint* $j^* \in B^\perp$ *with minimal slack given by*
$$\bar{s}_j := \hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j, \quad j \in B^\perp, \tag{5.57}$$
i.e.,
$$j^* := \arg\min_{j \in B^\perp} \{\bar{s}_j\}, \tag{5.58}$$
where $B^\perp \subset \mathbb{J}$ *contains all constraints, that are linearly independent with respect to the index set* B*.*

   (b) *Update the index set* B *and the set* $B^\perp \subset \mathbb{J}$*.*

After executing Algorithm 5.3, which performs basis crushing, we denote the basic solution corresponding to the basis B by $(\hat{x}, \hat{y})$. By construction
$$\hat{y}_k = \bar{y}_k \tag{5.59}$$
holds. Furthermore the $\bar{k}$-th row of the basis matrix $\hat{A}_B$ is equal to e_k, i.e.,
$$(\hat{A}_B)_{\bar{k}} = e_k. \tag{5.60}$$

After having constructed a basic solution $(\hat{x}, \hat{y})$ by Algorithm 5.3, we determine a suitable artificial constraint (5.12) by the subsequent algorithm. This artificial constraint is constructed in order to replace the artificial constraint (5.56) introduced in Algorithm 5.3 in order to ensure, that $(\hat{x}, \hat{y})$ is located in the correct disjunction, i.e., (5.7) holds.

Algorithm 5.4. *Let* $(\hat{x}, \hat{y})^0$ *be the basic solution with the corresponding basic set* B^0*, which is determined by Algorithm 5.3. Let* $\varepsilon > 0$ *be any tolerance for the minimal depth of a cutting plane. Therefore,* $|B^0| = n$ *and*
$$\hat{y}_k^0 = \bar{y}_k \tag{5.61}$$
hold, where $(\bar{x}, \bar{y})$ *is the optimal, non-basic solution of the continuous relaxation of MIQP* (5.2) *and* $\bar{y}_k$ *is fractional. Initialize artificial constraint* $(\hat{a}_a, \hat{b}_a)^0 := (\hat{a}_a^0, \hat{b}_a^0)$ *by*
$$(\hat{a}_a, \hat{b}_a)^0 := (e_k, \bar{y}_k). \tag{5.62}$$
Initialize the iteration index $l := 0$*.*

1. *Set* $l := l + 1$ *and define* $\hat{a}_a^l$ *as*

$$\hat{a}_a^l \;:=\; \left(e_{\bar{k}} + e_{\bar{k}}^{\perp} \left(\frac{1}{2} \right)^{l-1} \right)^T \hat{A}_{B^0} \tag{5.63}$$

where $e_{\bar{k}}^{\perp} \in \mathbb{R}^n$ *is given by*

$$(e_{\bar{k}}^{\perp})_i \;:=\; \begin{cases} 1, & i \in B^0 \backslash \{\bar{k}\}, \\ 0, & i = \bar{k}, \end{cases} \tag{5.64}$$

and $\bar{k}$ *corresponds to the position of the artificial constraint within* B.

2. *Define* $\hat{b}_a^l$ *by*

$$\hat{b}_a^l \;:=\; (\hat{a}_a^l)^T \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}^0. \tag{5.65}$$

3. *Include constraint* $(\hat{a}_a, \hat{b}_a)^l$ *in basis* B^l *at position* $\bar{k}$ *by removing* $(\hat{a}_a, \hat{b}_a)^{l-1}$.

4. *Check current disjunction and depth of simple disjunctive cut for* $(\hat{x}, \hat{y})^l$*:*

   If

$$\lfloor \bar{y}_k \rfloor \;\leq\; \hat{y}_k^l \;\leq\; \lceil \bar{y}_k \rceil, \tag{5.66}$$

   and

$$\pi^l \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \pi_0^l \;\leq\; -\varepsilon \tag{5.67}$$

   hold, where π *and* π_0 *define the simple disjunctive cut* (4.49) *induced by* $(\hat{x}, \hat{y})^l$*, see also* (4.50)*,* ***then RETURN.***

   Else ***GOTO*** *Step 1.*

The subsequent theorem proves, that Algorithm 5.4 terminates after a finite number of iterations.

Theorem 5.1. *Let* y_k *be fractional at the solution* $(\bar{x}, \bar{y})$ *of the continuous relaxation of MIQP* (5.2)*. Furthermore let the fractionality be greater than* $\sqrt{\varepsilon} > 0$*, i.e.,*

$$\lfloor \bar{y}_k \rfloor + \sqrt{\varepsilon} \;<\; \bar{y}_k \;<\; \lceil \bar{y}_k \rceil - \sqrt{\varepsilon} \tag{5.68}$$

holds. Let

$$\hat{a}_a \;:=\; \left(e_{\bar{k}} + e_{\bar{k}}^{\perp} \left(\frac{1}{2} \right)^{l-1} \right)^T \hat{A}_{B^0} \tag{5.69}$$

be linearly independent w.r.t. $\hat{A}_{B^0} \backslash \{e_k\}$.

Then Algorithm 5.4 terminates after a finite number $\hat{l}$ *of iterations satisfying conditions* (5.66) *and* (5.67) *for basis* $B^{\hat{l}}$ *containing the artificial constraint* $(\hat{a}_a, \hat{b}_a)^{\hat{l}}$ *and the corresponding basic solution* $(\hat{x}, \hat{y})^{\hat{l}}$.

Proof. Conditions (5.66) and (5.67) are satisfied, whenever Algorithm 5.4 terminates. As a consequence, we have to prove that updating rule (5.63) yields the desired artificial constraint $(\hat{a}_a, \hat{b}_a)^{\hat{l}}$.

Consider the limit of the iteration sequence given by

$$\lim_{l\to\infty} \hat{a}_a^l \;=\; e_{\bar{k}}^T \hat{A}_{B^0} \;=\; e_k, \tag{5.70}$$

which is denoted by iteration index l^*. This yields for $\hat{a}_a^{l^*} = e_k$ and

$$\hat{b}_a^{l^*} \;=\; (\hat{a}_a^{l^*})^T \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}^0 \;=\; e_k^T \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}^0 \;=\; \hat{y}_k^0 \;=\; \bar{y}_k. \tag{5.71}$$

Since

$$(\hat{A}_{B^{l^*}}^{-1})_k \;=\; e_{\bar{k}} \tag{5.72}$$

holds, we obtain

$$\hat{y}_k^{l^*} \;=\; \sum_{i\in B^{l^*},\ i\neq\bar{k}} (\hat{A}_{B^{l^*}}^{-1})_{ki} (\hat{b}_{B^{l^*}})_i + (\hat{A}_{B^{l^*}}^{-1})_{k\bar{k}} \hat{b}_a^{l^*} \;=\; \hat{b}_a^{l^*} \;=\; \bar{y}_k. \tag{5.73}$$

Therefore, condition (5.66) is satisfied for l^* due to

$$\lfloor \bar{y}_k \rfloor \;<\; \bar{y}_k \;<\; \lceil \bar{y}_k \rceil. \tag{5.74}$$

Exploiting (5.73), condition (5.67) is also satisfied for $(\hat{a}_c^{l^*}, \hat{b}_c^{l^*})$ obtained by assignment (4.53) from the simple disjunctive cut (4.49) corresponding to basis B^{l^*} given by

$$\pi \;:=\; \max\{\pi^1, \pi^2\} \;=\; (\bar{y}_k - \lfloor \bar{y}_k \rfloor) e_{\bar{k}} \tag{5.75}$$

and

$$\pi_0 \;:=\; (\lceil \bar{y}_k \rceil - \bar{y}_k)(\bar{y}_k - \lfloor \bar{y}_k \rfloor). \tag{5.76}$$

The reason is, that we obtain for the simple disjunctive cut given by

$$\pi^T s \;\geq\; \pi_0 \tag{5.77}$$

the following condition

$$\begin{aligned} \pi^T s - \pi_0 \;&=\; (\bar{y}_k - \lfloor \bar{y}_k \rfloor) e_{\bar{k}}^T s - (\lceil \bar{y}_k \rceil - \bar{y}_k)(\bar{y}_k - \lfloor \bar{y}_k \rfloor) \\ &=\; (\hat{y}_k - \lfloor \bar{y}_k \rfloor)\, e_{\bar{k}}^T \left(\hat{A}_{B^{l^*}} \begin{pmatrix} x \\ y \end{pmatrix} - \hat{b}_{B^{l^*}} \right) - (\lceil \bar{y}_k \rceil - \bar{y}_k)\,(\bar{y}_k - \lfloor \bar{y}_k \rfloor) \\ &=\; (\bar{y}_k - \lfloor \bar{y}_k \rfloor)(y_k - \bar{y}_k) - (\lceil \bar{y}_k \rceil - \bar{y}_k)(\bar{y}_k - \lfloor \bar{y}_k \rfloor). \end{aligned} \tag{5.78}$$

For $y_k = \bar{y}_k$ we get

$$\pi^T s - \pi_0 \;=\; -(\lceil \bar{y}_k \rceil - \bar{y}_k)(\bar{y}_k - \lfloor \bar{y}_k \rfloor) \;<\; -\sqrt{\varepsilon}\sqrt{\varepsilon} \;=\; -\varepsilon. \tag{5.79}$$

As a consequence, there exists an iteration index $\hat{l} < l^*$, where conditions (5.66) and (5.67) are satisfied, which proves the theorem. □

The requirement, that $\left(e_{\bar{k}} + e_{\bar{k}}^{\perp}\left(\frac{1}{2}\right)^{l-1}\right)^{T}\hat{A}_{B^0}$ is linearly independent w.r.t. the coefficient vectors of the constraints forming $\hat{A}_{B^0}\backslash\{e_k\}$, is no serious restriction. The vector $e_{\bar{k}}^{\perp}$ can simply be adapted, by setting some of its components in addition to zero, such that linear independence w.r.t. $\hat{A}_{B^0}\backslash\{e_k\}$ can be ensured.

Note, that, in every iteration l, the coefficients $\hat{a}_a^l$ are chosen by (5.63), such that $(\hat{x},\hat{y})^l$ is the unique optimal solution of the linear program (5.38) as shown in the subsequent corollary.

Corollary 5.7. *In every iteration l of Algorithm 5.4, the choice of the coefficients $\hat{a}_a^l$ determined by* (5.63) *ensure, that $(\hat{x},\hat{y})^l$ is the unique optimal solution of the linear program* (5.38)*.*

Proof. In each iteration l, the coefficients $\hat{a}_a^l$ are determined by

$$\hat{a}_a^l := \left(e_k + e_k^{\perp}\left(\frac{1}{2}\right)^{l-1}\right)^{T}\hat{A}_{B^0}. \tag{5.80}$$

As a consequence, the reduced costs generally given by

$$c_{B^\perp}^T - c_B^T\hat{A}_B^{-1}\hat{A}_{B^\perp} \tag{5.81}$$

are non-negative, since $c_{B^\perp}^T = 0$, $\hat{A}_{B^\perp} = -I$ and therefore

$$\begin{aligned} c_{B^\perp}^T - c_B^T\hat{A}_B^{-1}\hat{A}_{B^\perp} &= 0 - c_B^T\hat{A}_B^{-1}(-I) \\ &= (\hat{a}_a^l)^T\hat{A}_{B^0}^{-1} \\ &= \left(e_k + e_k^{\perp}\left(\frac{1}{2}\right)^{l-1}\right)^{T}\hat{A}_{B^0}\hat{A}_{B^0}^{-1} \\ &> 0 \end{aligned} \tag{5.82}$$

holds. This proves the corollary.

□

The subsequent algorithm efficiently constructs disjunctive cutting planes with strength at least ε for a non-basic solution $(\bar{x},\bar{y})$ or proves their non-existence.

Algorithm 5.5. *Let $\varepsilon > 0$ be an arbitrary minimal strength of the cut to be constructed. Furthermore, let y_k be fractional at the solution $(\bar{x},\bar{y})$ of the continuous relaxation of MIQP* (5.2)*, such that condition* (5.68) *holds.*

1. *Construct a basic solution $(\hat{x},\hat{y})$ by applying Algorithm 5.3 and Algorithm 5.4 and denote the corresponding basis by B.*

2. *Modify the right hand side* $\hat{b}_a$ *of the artificial constraint* $(\hat{a}_a, \hat{b}_a)$*, such that the simple disjunctive cut induced by* B *possesses depth* ε*:*

   If $(\hat{A}_B^{-1})_{k\bar{k}} < 0$, **then** *set* $\hat{b}_a$ *such that*

$$\begin{aligned} \rho - \rho_2^{<} - \rho_3\rho_4 + \rho_4^2 &= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j(\hat{b}_a)\left(\hat{a}_j^T\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} - \hat{b}_j\right) \\ &\quad + (\rho_4\bar{a}_{k\bar{k}} + \lceil\bar{y}_k\rceil\bar{a}_{k\bar{k}} - \rho_3\bar{a}_{k\bar{k}})\,\hat{b}_a \end{aligned} \tag{5.83}$$

   holds.

   Else *i.e.,* $(\hat{A}_B^{-1})_{k\bar{k}} > 0$*, set* $\hat{b}_a$ *such that*

$$\begin{aligned} \rho - \rho_2^{>} - \rho_3\rho_4 + \rho_4^2 &= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j(\hat{b}_a)\left(\hat{a}_j^T\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} - \hat{b}_j\right) \\ &\quad + (\rho_4\bar{a}_{k\bar{k}} + \lfloor\bar{y}_k\rfloor\bar{a}_{k\bar{k}} - \rho_3\bar{a}_{k\bar{k}})\,\hat{b}_a \end{aligned} \tag{5.84}$$

   is satisfied.

   Note, that the subsequent notation is introduced to ease the readability:

$$\rho := \varepsilon + \lceil\bar{y}_k\rceil\lfloor\bar{y}_k\rfloor, \tag{5.85}$$

$$\rho_2^{<} := -a_a^T\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}\bar{a}_{k\bar{k}}\lceil\bar{y}_k\rceil, \tag{5.86}$$

$$\rho_2^{>} := -a_a^T\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}\bar{a}_{k\bar{k}}\lfloor\bar{y}_k\rfloor, \tag{5.87}$$

$$\rho_3 := +a_a^T\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}\bar{a}_{k\bar{k}} + \lceil\bar{y}_k\rceil + \lfloor\bar{y}_k\rfloor, \tag{5.88}$$

$$\rho_4 := \sum_{j\in B,\ j\neq\bar{k}} (\hat{A}_B^{-1})_{kj}(\hat{b}_B)_j, \tag{5.89}$$

   and

$$\pi_j(\hat{b}_a) := \max\left\{(\rho_4 - \bar{a}_{k\bar{k}}\hat{b}_a - \lceil\bar{y}_k\rceil)(\hat{A}_B^{-1})_{kj}, (\rho_4 - \bar{a}_{k\bar{k}}\hat{b}_a - \lfloor\bar{y}_k\rfloor)(\hat{A}_B^{-1})_{kj}\right\}$$

   applying notation (4.25)*.*

3. *Compute the reduced costs* (4.55) *according to Step 2 of Algorithm 4.1.*

4. *Determine constraint* i^* *to be included in the basis according to Step 3 of Algorithm 4.1.*

5. *Determine constraint* j_* *to be removed from the basis according to Step 4 of Algorithm 4.1.*

6. *Execute basis exchange according to Step 5 of Algorithm 4.1.*

7. *Check if artificial constraint is still part of the simple disjunctive cut:*

   If *artificial constraint* $(\hat{a}_a, \hat{b}_a)$ *is still included in the current basis,* ***GOTO*** *Step 2.*

   Else *continue according to Algorithm 4.1, i.e.,* ***GOTO*** *Step 3.*

8. *Perturbation of row* k (4.86)*:*

   If *row* k *has no zero entries,* **then STOP**.

   Else *perturb row* k *by replacing every zero entry by* ε^t *for some small* $\varepsilon > 0$ *and* $t = 1, 2 \ldots$.

   If *the artificial constraint is still included in the current basis,* **then GOTO** *Step 2.*

   Else **GOTO** *Step 3.*

Note, that in the subsequent theorem we leave out the requirement needed in Theorem 5.1, that $\left(e_{\bar{k}} + e_{\bar{k}}^{\perp} \left(\frac{1}{2} \right)^{l-1} \right)^T \hat{A}_{B^0}$ is linearly independent w.r.t. the coefficient vector of the constraints forming $\hat{A}_{B^0} \backslash \{e_k\}$, since the vector $e_{\bar{k}}^{\perp}$ can easily be adapted to ensure linear independency as explained above.

Theorem 5.2. *Let* $(\bar{x}, \bar{y})$ *be the optimal non-basic solution of the continuous relaxation of MIQP* (5.2)*. Let* $n_a < n$ *linearly independent constraints be active at* $(\bar{x}, \bar{y})$ *and denote the corresponding index set by* B*. Let* e_k *be linearly independent w.r.t. the constraints contained in* B*. Let* $\varepsilon > 0$ *be any tolerance for the minimal depth of a cutting plane.*

Then Algorithm 5.5 either proves that no disjunctive cut exists with strength greater than ε *for the current disjunction* (4.27)*, or it constructs the disjunctive cut by solving the cut generating linear program* (4.45) *implicitly.*

Proof. To prove the theorem, we first show, that we obtain a simple disjunctive cut that satisfies

$$\pi^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \pi_0 \;=\; -\varepsilon, \tag{5.90}$$

if we modify the right hand side $\hat{b}_a$ of the artificial constraint $(\hat{a}_a, \hat{b}_a)$ according to (5.83) or (5.84), where the simple disjunctive cut is induced by basis B.

Then we show, that it is indeed possible in each iteration to choose the value of $\hat{b}_a$ as required. This is the case, if there is a value for $\hat{b}_a$, such that the simple disjunctive cut induced by the corresponding basis B has strength greater than or equal to ε and another value for $\hat{b}_a$, such that the simple disjunctive cut induced by B is weaker than ε.

Due to Corollary 5.4 the non-existence of disjunctive cutting planes stronger than ε is proved, if all reduced costs (4.55) are non-negative and the artificial constraint is still included in the current basis.

As soon as the artificial constraint $(\hat{a}_a, \hat{b}_a)$ is removed in Step 3, Algorithm 5.5 is equivalent to Algorithm 4.1 and therefore the disjunctive cut corresponding to the solution of the $CGLP_k$ is constructed.

Since the artificial constraint $(\hat{a}_a, \hat{b}_a)$ is modified in Step 3 in a monotone way, such that the strength of the cut is successively reduced, the procedure cannot cycle. Therefore the artificial constraint is either removed or the non-existence of disjunctive cuts stronger than ε is proved.

Now we go into detail:

Consider the basic solution $(\hat{x}, \hat{y})$, that was constructed by Algorithm 5.3 and Algorithm 5.4. By construction

$$\lfloor \hat{y}_k \rfloor \;=\; \lfloor \bar{y}_k \rfloor \tag{5.91}$$

holds.

First we show, that we obtain a cut that satisfies

$$\hat{a}_c^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_c \;=\; -\varepsilon, \tag{5.92}$$

if we modify the right hand side $\hat{b}_a$ of the artificial constraint $(\hat{a}_a, \hat{b}_a)$ according to (5.83) or (5.84).

Algorithm 5.4 yields a basis B and a corresponding basic solution $(\hat{x}, \hat{y})$. By construction an artificial constraint $(\hat{a}_a, \hat{b}_a)$ is included in B at position $\bar{k}$. The corresponding simple disjunctive cut (4.49) is given by

$$\begin{aligned} \pi_B^T \bar{s}_B &\geq \pi_0, \\ \pi_B^T \left(\hat{A}_B \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_B \right) &\geq \pi_0, \end{aligned}$$

with π_j, $j \in B$ and π_0 defined in (4.50). The aim is now to modify the simple disjunctive cut such that its strength is ε, i.e., the following condition is required

$$\begin{aligned} \pi_B^T \left(\hat{A}_B \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_B \right) - \pi_0 &= -\varepsilon \\ \sum_{j \in B,\; j \neq \bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j \right) + \pi_{\bar{k}} \left(\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_a \right) - \pi_0 &= -\varepsilon. \end{aligned} \tag{5.93}$$

Exploiting

$$\pi_0 \;:=\; (\lceil \bar{y}_k \rceil - \hat{y}_k)\,(\hat{y}_k - \lfloor \bar{y}_k \rfloor) \tag{5.94}$$

and

$$\begin{aligned} \pi_{\bar{k}} &:= \max\left\{\pi^1_{\bar{k}}, \pi^2_{\bar{k}}\right\} \\ &= \max\left\{(\hat{y}_k - \lceil \bar{y}_k \rceil)(\hat{A}_B^{-1})_{k\bar{k}}, (\hat{y}_k - \lfloor \bar{y}_k \rfloor)(\hat{A}_B^{-1})_{k\bar{k}}\right\} \end{aligned} \tag{5.95}$$

and

$$\begin{aligned} \hat{y}_k &= (\hat{A}_B^{-1}\hat{b}_B)_k \\ &= \sum_{j\in B,\ j\neq\bar{k}} (\hat{A}_B^{-1})_{kj}\hat{b}_j + (\hat{A}_B^{-1})_{k\bar{k}}\hat{b}_a, \end{aligned} \tag{5.96}$$

we obtain from (5.93)

$$\begin{aligned} \sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) &= -\varepsilon + \left(-\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{b}_a\right)\pi_{\bar{k}} + \pi_0 \\ &= -\varepsilon + \left(-\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{b}_a\right)\pi_{\bar{k}} \\ &\quad + (\lceil \bar{y}_k \rceil - \hat{y}_k)(\hat{y}_k - \lfloor \bar{y}_k \rfloor) \\ &= -\varepsilon + \left(-\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{b}_a\right)\pi_{\bar{k}} \\ &\quad + \lceil \bar{y}_k \rceil \hat{y}_k - \lceil \bar{y}_k \rceil \lfloor \bar{y}_k \rfloor - \hat{y}_k^2 + \lfloor \bar{y}_k \rfloor \hat{y}_k. \end{aligned} \tag{5.97}$$

Note, that the maximum in (5.95) is well-determined as $(\hat{A}_B^{-1})_{k\bar{k}}$ is known. By defining ρ according to (5.85) and exploiting (5.96), we obtain

$$\begin{aligned} \rho = &- \sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) + \left(-\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{b}_a\right)\pi_{\bar{k}} \\ &+ (\lceil \bar{y}_k \rceil + \lfloor \bar{y}_k \rfloor)\hat{y}_k - \hat{y}_k^2. \end{aligned} \tag{5.98}$$

First we assume, that $\pi_{\bar{k}} := \pi^1_{\bar{k}} = (\hat{y}_k - \lceil \bar{y}_k \rceil)(\hat{A}_B^{-1})_{k\bar{k}}$ holds, see (4.50), and apply

definition $\bar{a}_{k\bar{k}} =: -(\hat{A}_B^{-1})_{k\bar{k}}$, see (4.25). Then equation (5.98) yields

$$\begin{aligned}
\rho &= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) + \left(-\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{b}_a\right) \pi_{\bar{k}} + (\lceil \bar{y}_k \rceil + \lfloor \bar{y}_k \rfloor)\, \hat{y}_k - \hat{y}_k^2 \\
&= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) + \left(-\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{b}_a\right) (\hat{y}_k - \lceil \bar{y}_k \rceil)(-\bar{a}_{k\bar{k}}) \\
&\quad + (\lceil \bar{y}_k \rceil + \lfloor \bar{y}_k \rfloor)\, \hat{y}_k - \hat{y}_k^2 \\
&= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) + \left(-\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{b}_a\right) (-\bar{a}_{k\bar{k}})\hat{y}_k \\
&\quad + \left(-\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \hat{b}_a\right) \lceil \bar{y}_k \rceil \bar{a}_{k\bar{k}} + (\lceil \bar{y}_k \rceil + \lfloor \bar{y}_k \rfloor)\, \hat{y}_k - \hat{y}_k^2 \\
&= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) + \hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \bar{a}_{k\bar{k}}\hat{y}_k - \hat{b}_a \bar{a}_{k\bar{k}}\hat{y}_k \\
&\quad - \hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \bar{a}_{k\bar{k}} \lceil \bar{y}_k \rceil + \hat{b}_a \bar{a}_{k\bar{k}} \lceil \bar{y}_k \rceil + (\lceil \bar{y}_k \rceil + \lfloor \bar{y}_k \rfloor)\, \hat{y}_k - \hat{y}_k^2.
\end{aligned}$$

By defining $\rho_2^{\leq}$ according to (5.86), we obtain

$$\begin{aligned}
\rho &= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) + \hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \bar{a}_{k\bar{k}}\hat{y}_k - \hat{b}_a \bar{a}_{k\bar{k}}\hat{y}_k + \rho_2^{\leq} + \hat{b}_a \bar{a}_{k\bar{k}} \lceil \bar{y}_k \rceil \\
&\quad + (\lceil \bar{y}_k \rceil + \lfloor \bar{y}_k \rfloor)\, \hat{y}_k - \hat{y}_k^2 \\
&= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) - \hat{b}_a \bar{a}_{k\bar{k}}\hat{y}_k + \rho_2^{\leq} + \hat{b}_a \bar{a}_{k\bar{k}} \lceil \bar{y}_k \rceil \\
&\quad + \left(\hat{a}_a^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \bar{a}_{k\bar{k}} + \lceil \bar{y}_k \rceil + \lfloor \bar{y}_k \rfloor\right) \hat{y}_k - \hat{y}_k^2.
\end{aligned}$$

Simplification by defining ρ_3 according to (5.88) yields

$$\begin{aligned}
\rho &= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}_j\right) - \hat{b}_a \bar{a}_{k\bar{k}}\hat{y}_k + \rho_2^{\leq} \\
&\quad + \hat{b}_a \bar{a}_{k\bar{k}} \lceil \bar{y}_k \rceil + \rho_3 \hat{y}_k - \hat{y}_k^2.
\end{aligned} \tag{5.99}$$

By introducing ρ_4 according to (5.89), we obtain from (5.96) the condition

$$\hat{y}_k = \rho_4 - \bar{a}_{k\bar{k}} \hat{b}_a. \tag{5.100}$$

Together with the definition of π_j with $j \in B\backslash\{\bar{k}\}$ according to (4.50), we obtain

$$\begin{aligned}\pi_j(\hat{b}_\alpha) &:= \max\left\{(\hat{y}_k - \lceil \bar{y}_k \rceil)(\hat{A}_B^{-1})_{kj}, (\hat{y}_k - \lfloor \bar{y}_k \rfloor)(\hat{A}_B^{-1})_{kj}\right\} \qquad (5.101)\\ &= \max\left\{(\rho_4 - \bar{a}_{k\bar{k}}\hat{b}_\alpha - \lceil \bar{y}_k \rceil)(\hat{A}_B^{-1})_{kj}, (\rho_4 - \bar{a}_{k\bar{k}}\hat{b}_\alpha - \lfloor \bar{y}_k \rfloor)(\hat{A}_B^{-1})_{kj}\right\}\end{aligned}$$

Therefore we get

$$\begin{aligned}\rho &= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j \left(\hat{a}_j^T \begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} - \hat{b}_j\right) - \hat{b}_\alpha \bar{a}_{k\bar{k}}\hat{y}_k + \rho_2^< + \hat{b}_\alpha \bar{a}_{k\bar{k}}\lceil \bar{y}_k \rceil + \rho_3 \hat{y}_k - \hat{y}_k^2\\
&= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j(\hat{b}_\alpha) \left(\hat{a}_j^T \begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} - \hat{b}_j\right)\\
&\quad -\hat{b}_\alpha \bar{a}_{k\bar{k}}(\rho_4 - \bar{a}_{k\bar{k}}\hat{b}_\alpha) + \rho_2^< + \hat{b}_\alpha \bar{a}_{k\bar{k}}\lceil \bar{y}_k \rceil + \rho_3(\rho_4 - \bar{a}_{k\bar{k}}\hat{b}_\alpha) - (\rho_4 - \bar{a}_{k\bar{k}}\hat{b}_\alpha)^2\\
&= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j(\hat{b}_\alpha) \left(\hat{a}_j^T \begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} - \hat{b}_j\right)\\
&\quad -\hat{b}_\alpha \bar{a}_{k\bar{k}}\rho_4 + \bar{a}_{k\bar{k}}^2\hat{b}_\alpha^2 + \rho_2^< + \hat{b}_\alpha \bar{a}_{k\bar{k}}\lceil \bar{y}_k \rceil + \rho_3\rho_4 - \rho_3 \bar{a}_{k\bar{k}}\hat{b}_\alpha - \left(\rho_4^2 - 2\rho_4 \bar{a}_{k\bar{k}}\hat{b}_\alpha + \bar{a}_{k\bar{k}}^2\hat{b}_\alpha^2\right)\\
&= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j(\hat{b}_\alpha) \left(\hat{a}_j^T \begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} - \hat{b}_j\right)\\
&\quad +\rho_2^< + \rho_3\rho_4 - \rho_4^2 + \left(\rho_4 \bar{a}_{k\bar{k}} + \lceil \bar{y}_k \rceil \bar{a}_{k\bar{k}} - \rho_3 \bar{a}_{k\bar{k}}\right)\hat{b}_\alpha.\end{aligned}$$

For $\pi_{\bar{k}} := \pi_{\bar{k}}^2 = (\hat{y}_k - \lfloor \bar{y}_k \rfloor)(\hat{A}_B^{-1})_{k\bar{k}}$, see (4.50), we obtain analogously

$$\begin{aligned}\rho &= -\sum_{j\in B,\ j\neq\bar{k}} \pi_j(\hat{b}_\alpha) \left(\hat{a}_j^T \begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} - \hat{b}_j\right)\\
&\quad +\rho_2^> + \rho_3\rho_4 - \rho_4^2 + \left(\rho_4 \bar{a}_{k\bar{k}} + \lfloor \bar{y}_k \rfloor \bar{a}_{k\bar{k}} - \rho_3 \bar{a}_{k\bar{k}}\right)\hat{b}_\alpha\end{aligned}$$

with $\rho_2^>$ defined by (5.87). Note, that the maximum defining $\pi_j(\hat{b}_\alpha)$ in (5.101) is determined by the sign of $(A_B^{-1})_{kj}$ for $j \in B\backslash\{\bar{k}\}$, such that the evaluation of $\hat{b}_\alpha$ is straightforward.

Now we show, that it is indeed possible in each iteration to choose the value of $\hat{b}_\alpha$, such that equation (5.92) is satisfied. Note, that the cut given by $(\hat{a}_c, \hat{b}_c)$ in (5.92) is determined by the corresponding non-trivial, feasible, basic solution of the $CGLP_k$ induced by basis B containing the artificial constraint (5.12). As we showed above $\hat{b}_\alpha$ can in principle be chosen, such that (5.92) holds. We can ensure, that $\hat{b}_\alpha$ can be chosen appropriately, if the current cut is at least as strong as ε, i.e., condition (5.55) holds and if there is a value for $\hat{b}_\alpha$, such that the induced simple disjunctive cut has strength greater than or equal to ε and another value for $\hat{b}_\alpha$, such that the induced simple disjunctive cut is weaker than ε.

Due to the construction according to Algorithm 5.4, the initial cut possesses at least depth ε in Step 1. In every other iteration the strength is at least ε, since Step 3 improves the current cut, which already has strength ε. As a consequence, we can carry out Step 2, if there is always a weaker cut with strength less than ε. The subsequent calculations show, that this holds, if $\hat{b}_\alpha$ is set to either $\hat{b}^1_\alpha$ or $\hat{b}^2_\alpha$ specified below.

Consider the following two values for $\hat{b}_\alpha$

$$\begin{aligned} \hat{b}^1_\alpha &:= \frac{1}{(\hat{A}_B^{-1})_{k\bar{k}}} \left(\lceil \bar{y}_k \rceil - \sum_{i \in B,\ i \neq \bar{k}} (\hat{A}_B^{-1})_{ki} (\hat{b}_B)_i \right), \\ \hat{b}^2_\alpha &:= \frac{1}{(\hat{A}_B^{-1})_{k\bar{k}}} \left(\lfloor \bar{y}_k \rfloor - \sum_{i \in B,\ i \neq \bar{k}} (\hat{A}_B^{-1})_{ki} (\hat{b}_B)_i \right). \end{aligned} \tag{5.102}$$

Now define $\hat{b}_\alpha$ according to $\hat{b}^1_\alpha$, if

$$\hat{a}_\alpha^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}^1_\alpha \ > \ \hat{a}_\alpha^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}^2_\alpha \tag{5.103}$$

holds. Due to (5.96) $\hat{y}_k = \lceil \bar{y}_k \rceil$ holds, if $\hat{b}_\alpha$ is given by $\hat{b}^1_\alpha$, which yields the simple disjunctive cut induced by B

$$\begin{aligned} \pi_0 &= (\hat{y}_k - \lfloor \bar{y}_k \rfloor)(\lceil \bar{y}_k \rceil - \hat{y}_k) = (\lceil \bar{y}_k \rceil - \lfloor \bar{y}_k \rfloor)(\lceil \bar{y}_k \rceil - \lceil \bar{y}_k \rceil), \\ \pi_i &= \max\{\pi^1_i, \pi^2_i\}, \qquad \forall i \in B, \\ \pi^1_i &= (\lceil \bar{y}_k \rceil - \hat{y}_k)(\hat{A}_B^{-1})_{ki} = (\lceil \bar{y}_k \rceil - \lceil \bar{y}_k \rceil)(\hat{A}_B^{-1})_{ki}, \\ \pi^2_i &= (\hat{y}_k - \lfloor \bar{y}_k \rfloor)(\hat{A}_B^{-1})_{ki} = (\lceil \bar{y}_k \rceil - \lfloor \bar{y}_k \rfloor)(\hat{A}_B^{-1})_{ki}. \end{aligned} \tag{5.104}$$

Due to Lemma 4.1 the simple disjunctive cut given by (5.104) is not violated by $(\bar{x}, \bar{y})$, i.e., it possesses at most strength zero:

$$\pi^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \pi_0 \ \geq \ 0. \tag{5.105}$$

Furthermore, we define $\hat{b}_\alpha$ according to $\hat{b}^2_\alpha$, if condition

$$\hat{a}_\alpha^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}^2_\alpha \ > \ \hat{a}_\alpha^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \hat{b}^1_\alpha \tag{5.106}$$

is satisfied. Then the same consideration shows, that the simple disjunctive cut induced by B is not violated by $(\bar{x}, \bar{y})$, since $\hat{y}_k = \lfloor \bar{y}_k \rfloor$ holds.

Due to Corollary 5.4 the non-existence of disjunctive cutting planes stronger than ε is proved, if all reduced costs (4.55) are non-negative and the artificial constraint is still included in the current basis. If there are non-basic variables u_i or v_i with negative

reduced costs, Algorithm 5.5 yields an improving cut, which might still contain the modified, artificial constraint $(\hat{a}_a, \hat{b}_a)$.

As soon as the artificial constraint $(\hat{a}_a, \hat{b}_a)$ is removed in Step 3, Algorithm 5.5 is equivalent to Algorithm 4.1 and therefore the disjunctive cut solving $CGLP_k$ is constructed.

Since the artificial constraint $(\hat{a}_a, \hat{b}_a)$ is modified in Step 3 in a monotone way, such that the strength of the cut is successively reduced, the procedure cannot cycle. Therefore the artificial constraint is either removed or the non-existence of disjunctive cuts stronger than ε is proved.

□

6. NUMERICAL RESULTS

In this chapter the theoretical concepts and algorithms proposed in the previous chapters are evaluated. This chapter is divided into two sections. In the first part the performance of different MINLP solvers is compared. The algorithms are all part of a toolbox implemented at the University of Bayreuth. The second part of this chapter evaluates the performance of the MIQP solver MIQL executed with different settings. Furthermore MIQL is compared to the constraint programming solver SCIP, see Achterberg [2].

6.1 Comparative Study of MINLP Solution Methods

The MINLP solvers, which are analyzed in this section, are implementations of different algorithms for solving mixed-integer nonlinear optimization problems. The codes are part of a modular toolbox. Each is implemented in thread-safe Fortran as close to F77 as possible.

We evaluate the numerical performance of five solvers, some of them executed with alternative parameter settings, on a set of 100 academic mixed-integer test problems, collected by Schittkowski [95] and a set of 55 mixed binary test problems provided by our industrial cooperation partner Shell SIEP Rijswijk. Both test sets contain problems possessing a nonlinear and often also non-convex objective function. Furthermore, the feasible domains are non-convex in most test cases, and often nonlinear equality constraints are part of the problem description.

The following codes, based on the algorithms outlined in the previous chapters, are tested with different parameter settings:

MISQP [48]	-	Mixed-integer SQP-based trust region method, see Section 2.10.
MISQP/base [48]	-	Equivalent to MISQP, but quasi-Newton updates are not scaled and no restarts are performed, see Exler, Lehmann and Schittkowski [47] for further details.
MISQPOA [73]	-	Explicit combination of MISQP and a linear outer approximation method, i.e., additional stabilization by outer approximations, where successive mixed-integer nonlinear problems are solved by MISQP.

MIQPSOA [46]	-	Implementation of Algorithm 3.1, i.e., a MIQP-supported linear outer approximation method based on the successive solution of mixed-integer quadratic programs and linear outer approximation master problems.
MIQPSOA/OA [46]	-	Equivalent to MIQPSOA, but without mixed-integer search steps, i.e., equivalent to a linear outer approximation algorithm, see Section 2.5.
MINLPB4 [72]	-	NLP-based branch-and-bound method, where the continuous nonlinear programs are solved by MISQP, see Section 2.4.

For solving continuous quadratic programming problems, we use an extension of the code QL of Schittkowski [94], which is based on an implementation of Powell [87]. The underlying primal-dual method of Goldfarb and Idnani [58] was described in Section 2.11 and is extended in order to perform warmstarts. All convex mixed-integer quadratic programs are solved by the code MIQL, see Lehmann and Schittkowski [71], which is a cut-and-branch solver based on continuous quadratic relaxations.

All continuous nonlinear programs, e.g., arising in the NLP-based branch-and-bound approach are solved by MISQP setting the number of integer variables to zero. In this case the algorithm behaves like an SQP algorithm with trust region stabilization and quasi-Newton updates, see Exler et al. [48] for details. Note, that the same holds for MIQPSOA, see Section 3.1.

The mixed-integer quadratic programming code MIQL and the NLP-based branch-and-bound method MINLPB4 use the branch-and-bound framework BFOUR, see Lehmann et al. [74], where several branching and node-selection strategies are implemented, see also Section 6.2.

For executing the above mentioned optimization codes, we apply default parameter settings and tolerances, see the corresponding user guides for details, with a termination tolerance 10^{-6}. The maximal number of iterations is 2,000, and the number of branch-and-bound nodes is bounded by 10,000. All test examples are provided with the best-known objective function value $f^\star$, which is either obtained from analytical solutions, literature, or extensive numerical testing.

To be as close as possible to complex practical engineering applications, derivatives with respect to continuous variables are always approximated by forward differences subject to a small tolerance of 10^{-6}. Gradients for integral variables are approximated at neighbored grid points, in order to deal with non-relaxable problems, i.e., problems where the model functions cannot be evaluated at a point where some integer variables possess fractional values. For binary variables or for integer variables at a bound, a forward or backward difference formula is applied, respectively, see Exler, Lehmann and Schittkowski [47] for details. These descent directions might be considered as a very crude numerical approximation of partial derivatives by finite differences.

The Fortran codes are compiled by the GFORTRAN Compiler under SUSE Linux and the test runs are executed on an Intel Core(TM)2 Duo P8700 64 bit processor

with 2.53 GHz and 4 GB RAM.

The subsequent criterion defines a successful test run. Let $\epsilon_t > 0$ be a tolerance for defining the relative accuracy, and (x^k, y^k) the final iterate of a test run. To be able to evaluate test cases with $f^\star = 0$ with respect to (6.1), as in some of the academic test instances, we add the value one to the objective function. We call (x^k, y^k) a *successful* solution, if the relative error in the objective function is less than ϵ_t and if the maximum constraint violation is less than ϵ_t^2, i.e., if

$$f(x^k, y^k) - f^\star < \epsilon_t |f^\star| \tag{6.1}$$

and

$$\|g(x^k, y^k)^-\|_\infty < \epsilon_t^2, \tag{6.2}$$

where $g(x^k, y^k)^-$ represents the vector of constraint violations defined by (2.29). The tolerance for measuring the constraint violation is chosen to be smaller than the tolerance for the error in the objective function. In the latter case we apply a relative measure, whereas the constraint functions of our test problems are often badly scaled.

Moreover, we would like to distinguish three kinds of test results. Successful test runs satisfying (6.1) and (6.2), termination with an acceptable solution, which is feasible, but non-optimal, and false terminations. A test run ends with a false termination, if a error message is returned or if no feasible solution could be found. A solution (x^k, y^k) is called *acceptable*, if (6.2) holds and if the internal termination conditions subject to a reasonably small tolerance of $\epsilon = 10^{-6}$ are satisfied. Furthermore we require

$$f(x^k, y^k) - f^\star \geq \epsilon_t |f^\star| \tag{6.3}$$

instead of (6.1). For our numerical tests, we use $\epsilon_t = 0.01$.

We use the subsequent criteria to compare the robustness and efficiency of our codes:

- n_{succ} - number of successful test runs
- n_{acc} - number of acceptable, i.e., of non-optimal feasible solutions
- n_{err} - number of test runs terminated by an error message or without a feasible solution
- Δ_{err} - average relative deviation of computed solution from the best known one, i.e., $(f(x^k, y^k) - f^\star)/|f^\star|$
- n_{func} - average number of equivalent function calls including function calls used for computing a descent direction or gradient approximations, evaluated over all successful test runs, where one function call consists of one evaluation of the objective function and all constraint functions
- $time$ - average execution times in seconds, evaluated over all successful test runs

Table (6.1) provides details on all parameters used for executing the implementation of MIQPSOA, which is proposed in Chapter 3:

Δ_c^0	-	initial trust-region radius for continuous variables	10^1
Δ_i^0	-	initial integer trust-region radius for integer variables	10^1
σ^0	-	initial penalty parameter	10^1
δ^0	-	initial parameter for penalty parameter update	10^{-2}
ε_{OA}	-	tolerance for linear outer approximation master problem	10^{-4}
ε	-	tolerance for ε-stationary points	10^{-5}
$\bar{\sigma}$	-	upper bound on penalty parameter	10^{10}

Tab. 6.1: Initial Parameter-Setting for MIQPSOA

6.1.1 Academic Test Problems

First, the performance of the presented solvers is evaluated on a test set of 100 academic test examples published by Schittkowski [95]. For each test problem the best-known objective function value is provided. It has either been found in the literature or it has been obtained by extensive testing over several years.

The maximal number of variables within the test set is 23 for continuous, 100 for integer, and 24 for binary variables. Moreover, there are up to 17 equality constraints and the total number of constraints is at most 75. 65 test problems are taken from the GAMS MINLPLib, see Bussieck, Drud, and Meeraus [34]. Table 6.2 shows numerical results obtained for the mixed-integer nonlinear solvers MISQP, MINLPB4, MISQPOA and MIQPSOA, see the previous subsection for more details.

In a few cases, the codes cannot find an acceptable solution or an error message is generated. In a couple of other situations, the codes are unable to improve a current iterate and report that an acceptable solution is obtained, which is not the global optimum.

code	n_{succ}	n_{acc}	n_{err}	Δ_{err}	n_{func}	*time*
MISQP	90	10	0	0.375	897	2.1566
MISQP/base	51	48	1	19.940	612	0.5267
MISQPOA	89	9	2	1.769	1,000	0.6894
MIQPSOA	74	24	2	0.837	1,274	2.7077
MIQPSOA/OA	62	27	11	2.143	1,264	2.9699
MINLPB4	86	8	6	0.907	9,480	0.0768

Tab. 6.2: Performance Results for a Set of 100 Academic Test Problems

Table 6.2 shows that MISQP using default settings is the most reliable solver, since it finds the best-known solution in 90 per cent of all test cases. Furthermore, we observe that fine-tuning is very crucial, since the reliability decreases significantly, if the quasi-Newton matrix is not scaled and no restarts are performed. This basic

version can only find the best-known solution in about half of the test cases. MISQP is very efficient in terms of the number of function evaluations, which is the most important performance criterion for simulation-based optimization problems. Only the basic version need slightly less function calls, which is cased by early termination at non-optimal iterates.

MISQPOA calls MISQP within an outer approximation framework. Thus, the obtained solution is at least as good as the one found by MISQP, but the number of function evaluations is higher due to the extra efforts to validate global optimality. Global optimality can be guaranteed, if the problem is convex and exact gradients are provided to formulate the outer approximation master problem (2.75).

The outer approximation algorithm called MIQPSOA is less reliable than MISQP using default settings, but the reliability is much higher than that of MISQP without fine-tuning, i.e., scaling of the quasi-Newton matrix and restarts, see MISQP/base. These successful features can in principle be included in MIQPSOA, such that its reliability can be further improved. As shown in Chapter 3, global optimal solutions are obtained, if the MINLP is convex and exact gradients can be provided for the outer approximation master problem (2.75), see Theorem 3.1. The average number of function calls of MIQPSOA is significantly higher compared to MISQP, since the algorithm terminates only if the master problem (2.75) is infeasible.

The difference between well-known linear outer approximation algorithms, e.g., DICOPT [43], and MIQPSOA are the mixed-integer search steps in order to obtain improving mixed-integer search directions according to Definition 3.2. Table 6.2 shows, that the reliability is significantly increased by performing mixed-integer search steps, while the efficiency in terms of the number of function evaluations is comparable.

As expected, the NLP-based branch-and-bound solver MINLPB4 is much less efficient in terms of the number of function evaluations, since a large number of continuous nonlinear optimization problems must be solved. Nevertheless, it is very reliable, even if it is applied to non-convex problems.

6.1.2 Test Problems from Petroleum Engineering

A large variety of applications of mixed-integer nonlinear programming is found in the petroleum industry. We select two classes of problems known as well relinking and gas lift problems for our numerical tests. The test cases differ in their dimensions and data and are collected in a test set containing 55 MINLPs. These applications are based on complex simulators, but simplified algebraic description are provided by Shell SIEP Rijswijk reproducing typical problem characteristics. For each test case the value of the best-known solution was also reported, which was either found by extensive numerical tests or by global optimization solvers.

To give an example, we introduce a simple well relinking model, where the total flow in a given network is to be maximized. The network consists of a certain number of

source nodes and some sink nodes, see Figure 1.1 for a typical example. The flow from each source node is to be directed to exactly one sink node, and the total capacity at the sinks is limited in terms of pressure and flow. A source node has a special pressure-flow characteristic and the total flow within the network is bounded. The pressure flow interactions are modeled by nonlinear functions, whereas the network is represented by binary variables.

Let us assume that there are m_s sinks and n_s sources, and that we want to maximize the total flow

$$\sum_{i=1}^{n_s} x_i$$

under so-called split-factor constraints, i.e., a set of switching conditions for each source i, $i = 1, \ldots, n_s$, of the form

$$\sum_{j=1}^{m_s} y_j^i \;=\; 1 \;.$$

Moreover, we have pressure constraints at source i, $i = 1, \ldots, n_s$,

$$\sum_{j=1}^{m_s} c_j^i y_j^i \;\leq\; a_i - b_i x_i \;,$$

and upper bounds Q_j for mass rates at the sinks, $j = 1, \ldots, m_s$,

$$\sum_{i=1}^{n_s} x_i y_j^i \;\leq\; Q_j \;,$$

with appropriate positive constants c_j^i, Q_j, $j = 1, \ldots, j = m_s$, and a_i, b_i, $i = 1, \ldots, n_s$. The well relinking test examples are defined by their dimensions $m_s = 3$ and $n_s = 3$, $n_s = 6$, or $n_s = 9$, the constants mentioned above, modified topologies, and the existence of simulated compressors and related technical systems or not.

In a very similar way, some gas lift test problems are created, see Ray and Sarker [91] or Ayatollahi et al. [8] for related models. We finally get a set of 55 test problems, where the number of continuous variables varies between 3 and 10, the number of binary variables between 9 and 27, the number of linear equality constraints between 0 and 9, and the number of inequality constraints between 1 and 21. Table 6.3 contains performance results for the solvers under consideration.

The results differ not too much from those presented in the last subsection for the academic test set. The code MISQP without fine-tuning is by far the most efficient one in terms of number of function evaluations. Since the fine-tuning features are missing its reliability is much lower than that of MISQP using default settings. The optimal solution is only found in less than 60 percent of the test cases, while the best-known solution is obtained for 51 out of 55 problems, if fine-tuning is turned on. Although the

code	n_{succ}	n_{acc}	n_{err}	Δ_{err}	n_{func}	*time*
MISQP	51	3	1	0.0451	2,097	1.0377
MISQP/base	32	22	1	0.1114	545	0.1331
MISQPOA	52	3	0	0.0245	17,827	11.3544
MIQPSOA	39	16	0	0.0887	9,680	3.6311
MIQPSOA/OA	39	15	1	0.1801	29,028	19.4237
MINLPB4	55	0	0	0.0	157,340	0.5834

Tab. 6.3: Performance Results for a Set of 55 Test Problems from Petroleum Industry

average number of function evaluations increases by a factor of almost four, MISQP is still much more efficient than the other solvers. Even if an optimal solution is not reached, MISQP stops at least at an acceptable solution.

The additional stabilizations of MISQPOA by linear outer approximations require a significant amount of additional iterations and, consequently, a much larger number of function calls. The reliability is only slightly improved yielding the best-known solution in almost 95% of the test cases. For non-optimal solutions the average error between the best-known solution and the acceptable solution is reduced by a factor of almost 2.

The new outer approximation code MIQPSOA is less reliable than MISQP using fine-tuning, but again the number of successful test runs is significantly higher than that of MISQP/base. MIQPSOA need about 5 times as many function calls as MISQP. Compared to well-known outer approximation methods, i.e., MIQPSOA/OA, the number of function calls is reduced by a factor of 3, while the reliability is comparable.

The branch-and-bound code MINLPB4 solves all test problems, but requires more than 75 times as many function calls as MISQP. Nevertheless the average computation time is very low.

6.2 Solving Convex MIQP Problems

Our main motivation for developing a MIQP solver is to solve the subproblems, arising in MIQP-based MINLP solvers, efficiently. These problems possess a dense objective function, since the quadratic objective is derived by a quasi-Newton update, see Definition 2.7. The constraints are linearizations of general, unstructured, nonlinear constrains. Due to the algorithmic set up of the MIQP-based solution methods, the problems are feasible and an integer feasible solution is always available by construction.

To be able to evaluate the performance of the MIQP solver for those problems, we developed a test-framework and a library of test-cases. These problems are subproblems, which arise during the solution of the MINLP problems described in Section 6.1. We do not consider problems that are solved within half a second, since all available

solution strategies perform sufficiently well and differences are hard to measure.

6.2.1 Survey of Algorithmic Settings for MIQP Solvers

There are plenty of different settings available in state-of-the-art MILP solvers, see e.g., Achterberg [4] for the main parameters of the MILP solver SCIP. In principle the same settings are associated with MIQP solvers as well. In this section we provide a very brief review on some of these settings, that are available in our MIQP solver MIQL. In addition we mention some well-known alternatives, which are not yet implemented. The main decision of branch-and-bound solvers for MILP and MIQP problems is the selection of the branching and the node-selection rule. Furthermore the cut generation management and the integration of heuristics plays a crucial role.

For all test runs we apply the same branching strategy, which is called maximum fractional branching. Extensive research was carried out on branching strategies for MILP problems, see e.g., Achterberg, Koch and Martin [3]. As a result, many different branching rules were proposed such as strong branching, inference branching or reliability branching. Strong branching results in a small search tree, but the computational effort is high, since a trial branching is carried out for each fractional integer variable at each node. Inference branching turned out to be especially helpful for solving feasibility problems, as it does not rely on information of the objective function. It is based on information retrieved from domain propagation instead. If no further problem specific information can be exploited, reliability branching is a very successful branching rule. It is an enhancement of pseudo-cost-branching, which measures the improvement caused by branching on a certain variable with respect to the objective function in dependence of the fractionality of the variable. This information is either collected from previous branchings or obtained by strong branching, if historical data is not reliable, see Achterberg [4] for more details. For our MIQP solver those branching rules are not available yet and therefore we have to apply maximum fractional branching. As a consequence, computation times might be further reducible by implementing more advanced branching strategies.

Compared to the research related to branching rules, node-selection is a less elaborated topic, but recently new research was initiated, see e.g., Wojtaszek [110]. There are two common strategies for selecting the node to proceed in the next iteration of the branch-and-bound enumeration. The first one is *best-first-search*, where the next node is always the one with the lowest dual bound. As a consequence, the dual bound is improved quickly but successive subproblems are usually unrelated and therefore warmstarts can hardly be performed. In addition, primal solutions are found very late such that bounding cannot be applied. This leads to large number of open nodes in the search tree. This effect is strengthened since infeasible subproblems are rarely encountered. Therefore, a large amount of storage is required.

The second class of node-selection strategies is *depth-first-search*. In pure depth-first-search one of the successive nodes is chosen whenever possible. If the current node is

a leaf, i.e., it is integral, infeasible or dominated by the current upper bound the subsequent node is chosen via backtracking in the neighborhood of the current node. The advantages are, that all subproblems are closely related and therefore warmstarting techniques can be successfully applied to reduce the computational effort. Moreover, the number of unexplored nodes is low, which reduces the storage requirements drastically. Furthermore, primal solutions are usually found fast, such that bounding can be applied early. Compared to best-first-search more subproblems need to be solved, since the dual bound is not taken into account. A modification of this node-selection rule is called *depth-first-search with restarts*. Whenever a leaf is encountered the subsequent node is chosen to be the open node in the search tree possessing the lowest dual bound. As a consequence, the dual bound is also improved successively but the advantages of depth-first-search are maintained. There are node-selection strategies, that try to find good nodes by considering the fractionality of the integer variables and the dual bound, see Achterberg [4] for further details.

Warmstarts for continuous quadratic programs were described in detail in Section 2.11. The efficiency of warmstarts is closely related to the chosen node-selection strategy, as pointed out above. The speed-up caused by warmstarts is evaluated in the subsequent section.

Basic theory of cutting planes was reviewed in Chapter 4. Furthermore, a method for efficiently generating cutting planes for non-basic solutions was developed in Chapter 5. Corresponding results verifying the efficiency of the cut generator and show, that a significant speed-up can be obtained by constructing cutting planes for some instances, see Section 6.2.2.

The possibility of fathoming a node is determined by the quality of the current upper bound and the dual bound at the node. If the dual bound can be strengthened, the corresponding node might be cut off earlier. If the continuous quadratic subproblems are solved by a dual method, e.g., the one proposed by Goldfarb and Idnani [58], see Section 2.11, each iteration increases the dual bound. As a consequence, one can carry out the branching subject to the selected branching rule and perform a small number of additional iterations for both created child nodes. Afterwards, the value of the objective function is a valid dual bound, which might be much more stringent than the objective value at the parent node. The strengthening of the dual bound by additional iterations of a dual QP solver was proposed by Leyffer and Fletcher [75].

Due to the efficiency of warmstarts depth-first-search is one preferable node-selection strategy. The drawback is, that diving with a random choice of the successive node, results in a drastic increase of the branch-and-bound nodes. A possible alternative is the node-selection rule *best-of-two*, which performs a depth-first-search but solves both child problems and chooses the better. Best-of-two reduces the number of branch-and-bound nodes significantly compared to pure depth-first-search and warmstarts can be exploited. A further alternative is *depth-first-search with restarts guided by the Lagrangian function*, where both child nodes of the current node are inspected by evaluating the Lagrangian function $L(x, y, \lambda, \mu)$, see Definition 2.4, at $\lfloor \bar{y}_i^k \rfloor$ and $\lceil \bar{y}_i^k \rceil$.

$(\bar{x}^k, \bar{y}^k, \bar{\lambda}^k, \bar{\mu}^k)$ is the solution of the k-th branch-and-bound subproblem and y_i is the chosen branching variable. If

$$L(\bar{x}^k, \tilde{y}^k, \bar{\lambda}^k, \bar{\mu}^k) < L(\bar{x}^k, \hat{y}^k, \bar{\lambda}^k, \bar{\mu}^k) \tag{6.4}$$

with

$$\tilde{y}^k := \begin{cases} \bar{y}_j^k, & \forall j \in \{1, \ldots, n_i\} \backslash \{i\} \\ \lfloor \bar{y}_i^k \rfloor, & \text{otherwise} \end{cases} \tag{6.5}$$

and

$$\hat{y}^k := \begin{cases} \bar{y}_j^k, & \forall j \in \{1, \ldots, n_i\} \backslash \{i\} \\ \lceil \bar{y}_i^k \rceil, & \text{otherwise} \end{cases} \tag{6.6}$$

holds, the node with $y_i \leq \lfloor \bar{y}_i^k \rfloor$ is chosen. Otherwise, we proceed with the node containing the branching condition $y_i \geq \lceil \bar{y}_i^k \rceil$. Among all inspected node-selection strategies depth-first-search with restarts guided by the Lagrangian function is the most efficient enumeration scheme. It was proposed by Nowak [83].

6.2.2 Numerical Results for MIQP Solver MIQL

We analyze the performance of the solver MIQL, see Lehmann et al. [71], with different settings varying the node-selection strategy, the number of maximal successive warm-starts and the construction mode for cutting planes. Furthermore, dual information can be used to improve the dual bound of unexplored nodes and the node-selection during depth-first-search can be guided by the value of the Lagrangian function.

The solver MIQL is implemented in thread-safe Fortran as close to F77 as possible. The framework BFOUR is applied to perform the branch-and-bound enumeration, see Lehmann et al. [74]. The continuous quadratic subproblems are solved by an extension of the quadratic solver QL [94], see also Section 2.11.

We evaluate the performance in terms of total calculation time and number of branch-and-bound nodes. The solution provided by MIQL is the global optimum of the convex MIQP (2.127). In case of numerical errors caused by the quadratic solver QL, we continue with the valid lower bound given the last iteration point of the method of Goldfarb and Idnani [58], see Section 2.4 and Section 2.11 for further details. MIQL is executed with default settings apart from the options that are to be evaluated, see the corresponding user's guide [71].

The test set consists of 46 MIQP problems. They arise as subproblems during the solution of 8 from the 100 academic test cases collected by Schittkowski [95] by the solver MISQP, see Section 2.10. The number of variables is in the range of 21 to 51, while the number of integral variables varies from 20 to 50. The test cases possess up to 55 constraints. The 46 MIQP problems are exactly those subproblems, which cannot be solved to optimality within 1 second.

Setting	Node Sel.	Impr. Bnds	Lagr. Dir.	Warmstarts	DJ-Cuts
$MIQL_1$	Depth-First	—	yes	yes	no
$MIQL_2$	Depth-First	—	yes	yes	standard
$MIQL_3$	Depth-First	—	yes	yes	efficient
$MIQL_4$	Depth-First	—	yes	no	no
$MIQL_5$	Best-First	yes	—	yes	no
$MIQL_6$	Best-First	yes	—	yes	efficient
$MIQL_7$	Best-First	no	—	yes	no
$MIQL_8$	Depth-First	—	no	yes	no

Tab. 6.4: MIQL Settings

Table 6.4 shows the different settings that are inspected.

The main performance criterion is the calculation time needed to find the global solution of the convex MIQP (2.127). Another interesting characteristic is the number of branch-and-bound nodes. In principle, these two criteria are closely related but the ability to perform warmstarts leads to a variation. Table 6.5 shows the performance of the different MIQL settings for the 46 test problems. The calculation time in seconds and the number of branch-and-bound nodes are compared by considering the geometric mean for all test cases. Furthermore, we report the average gap in percent, that is closed, if the construction of cutting planes yields at least one cut. The closed gap is defined to be the difference between the objective value of the continuous relaxation with and without cutting planes relative to the optimal objective value.

Finally we report the average number of quadratic subproblems, which could not be solved to optimality due to internal numerical errors, e.g., division by a very small number. Note, that a valid lower bound is still available, if a numerical error occurs during the solution of a continuous quadratic subproblem by the method of Goldfarb and Idnani [58]. Therefore, the optimality of the solution can nevertheless be guaranteed. Since the number of branch-and-bound nodes might be increased due to the weaker lower bound at those nodes, numerical errors occurring during the solution of the continuous quadratic subproblems can increase the computation time. We obtain disjunctive cutting planes in 25 out of 46 test cases at the root node.

Table 6.5 shows that combining depth-first-search with restarts guided by the value of the Lagrangian function at $\lfloor \bar{y}_i^k \rfloor$ or $\lceil \bar{y}_i^k \rceil$ respectively, is beneficial. Here $\bar{y}_i^k$ is the branching variable at the solution $(\bar{x}^k, \bar{y}^k)$ of the k-th quadratic subproblem. Comparing setting $MIQL_1$ and $MIQL_8$ we see, that the computation time with respect to the geometric mean is reduced by more than 25 percent. The main reason for this improvement is the reduction of the number of branch-and-bound nodes, which is reduced by more than 33 % with respect to the geometric mean.

The computation of improved bounds as proposed by Leyffer and Fletcher [75] turns out to be beneficial as well. Considering settings $MIQL_5$ and $MIQL_7$ the number of branch-and-bound nodes is reduced by 18 %, which yields a reduction of the compu-

Setting	Time	No. Nodes	Closed Gap	QP failures
$MIQL_1$	13.18	42033.17	—	42.00
$MIQL_2$	15.91	37979.85	6	69.22
$MIQL_3$	13.53	37637.26	7	92.13
$MIQL_4$	18.18	40971.50	—	77.98
$MIQL_5$	18.22	28881.80	—	58.87
$MIQL_6$	16.94	26250.76	7	99.13
$MIQL_7$	23.80	35232.33	—	76.09
$MIQL_8$	17.74	63167.55	—	68.59

Tab. 6.5: MIQL Results

tation times of 23 percent, due to shaper lower bounds.

Applying warmstarts turns out to be efficient in combination with the node-selection strategy depth-first. Comparing $MIQL_1$ and $MIQL_4$ shows, that the computation time is reduce by more than 27 %. The number of QPs, where numerical errors arise is even reduced by almost 47%. As expected warmstarts do not have a significant influence on the number of branch-and-bound nodes.

Although disjunctive cutting planes only exist in about 54 % of the test cases, the efficient construction method proposed in Chapter 4 reduces the computation time by more than 7 percent in combination with best-first-search, comparing $MIQL_5$ and $MIQL_6$. Combining the efficient construction of cutting planes with depth-first-search guided by the value of the Lagrangian function yields a slight increase of the computation time of about 2 percent, as we can conclude by considering $MIQL_1$ and $MIQL_3$. The construction of cutting planes leads to a significant increase of subproblems, where numerical problems arise. Furthermore, the number of branch-and-bound nodes is reduced by more than 10 %. Comparing the cut generation modes, i.e., $MIQL_2$ and $MIQL_3$, shows that the efficient construction method for disjunctive cutting planes reduces the total calculation time by almost 15 %. Since the effort for constructing the cutting planes is reduced significantly, it does not matter, that disjunctive cutting planes do not exist at the root node in almost 46% of the test cases.

Analyzing the results in more detail, see Table B.2 in Appendix B, we see that the generation of cutting planes has a significant influence on the calculation time. As expected the calculation time is related to the number of branch-and-bound nodes. The generation of disjunctive cutting planes leads to a reduction of the number of nodes in two thirds of all test cases, where disjunctive cutting planes were found at the root node. Considering the geometric mean, the number of branch-and-bound nodes is reduced by almost 30 %, which corresponds to a reduction of the calculation time of 20 percent in these test cases. The maximal reduction of nodes is 95 percent, which reduces the calculation time by more than 80 %.

In one third of the test cases, where disjunctive cuts were found at the root node, the number of branch-and-bound nodes is increased by almost 50 %. As a consequence

the calculation times double. Considering the maximal increase, three times as many branch-and-bound nodes are needed and the calculation time is increased by a factor of almost 3.

As a consequence, further research on the construction of cutting planes for MIQPs is necessary, figuring out under which conditions they are beneficial. Their generation might be more advantageous, if the reliability of the QP solver is improved, since a significant increase of numerically difficult subproblems is detected. Therefore, an active cut management during the solution process is needed, in order to improve the numerical stability of the quadratic problem by removing instable cutting planes at least temporarily. A cut management could further enhance the performance, since cutting planes could be constructed at nodes, that are located deeper in the search tree. The probability, that disjunctive cuts exit, increases on lower levels of the branch-and-bound tree, since more constraints are part of the corresponding QP, see the proof of Lemma 5.4.

In addition, we also compare the performance of MIQL with the constraint programming framework SCIP, see Achterberg [2]. SCIP applies various presolving techniques and cut generators, see Wolters [111]. Furthermore, a large number of primal heuristics is implemented for finding primal solutions quickly, see Berthold [23]. SCIP is applied in many research projects, e.g., for topology planning in gas transportation networks, see Fügenschuh et al. [55]. SCIP is able to solve a large variety of problem types from MILPs to non-convex MIQCPs and special non-convex MINLP problems. A comparative study on solving MIQCP problems is available, see Berthold, Heinz and Vigerske [24].

SCIP is executed on the 46 test instances with standard settings using the latest version 2.0.2 with all available nonlinear features. SCIP is able to solve 31 out of 46 problems with a time-limit of 1000 seconds. The average optimality gap, i.e., the relative difference between the best dual and the best primal bound is 39 % for those test cases, that could not be solved to optimality. The geometric mean of the solution time for all problems is 32.33 seconds.

Neglecting all instances, that were not solved within the time limit, yields a different result. For the remaining test cases the geometric mean of SCIP is 6.08 seconds, while the fastest MIQL setting, i.e., MIQL_3, has a geometric mean of 13.33 seconds. Furthermore, the differences in the solution times are huge. There are instances, that SCIP can solve in less than one second, while MIQL needs more than 935 seconds. For other test cases MIQL can determine the optimal solution in less than 16 seconds, while SCIP reaches the time-limit of 1000 seconds and the optimality gap, i.e., the relative difference of the primal and dual bound, is still larger than 60 %.

Analyzing the solver information of SCIP shows, that presolving is not very successful, since only few bounds can be tightened. It is not possible to remove variables in any instance and only sometimes a single constraint is redundant. Furthermore, only few cutting planes can be found, mostly complemented mixed-integer rounding cuts and sometimes flowcover cuts.

A careful inspection shows, that the solutions of SCIP is worse than those of MIQL in 7 out of 31 test cases. In 5 of these problems the derivation is less than 4 percent. In the remaining two problems the solution of SCIP is 30 % and 59% worse than that of MIQL. All solutions satisfy the constraints with respect to the feasibility tolerance, which is 1.0E-6 for SCIP and 1.0E-10 for MIQL.

The detailed results of all 46 MIQP test cases are presented in the Appendix B.

7. CONCLUSION

This dissertation deals with mixed-integer nonlinear optimization problems. The main focus was the development of new solution methods that are very efficient in terms of the number of function evaluations, that are needed to solve MINLP problems. If the number of function evaluations is sufficiently low, the corresponding algorithm can be applied to solve simulation-based optimization problems, that rely on complex simulation tools, which frequently arise in industrial applications. Since available methods, especially NLP-based branch-and-bound methods, need a large number of function evaluations, they cannot be applied, if function evaluations are expensive.

After reviewing available solution techniques for convex MINLP problems, we propose a new outer approximation algorithm that employs additional mixed-integer search steps, denoted by MIQPSOA. For this MINLP algorithm we prove convergence for convex mixed-integer nonlinear programs using standard assumptions, e.g., exact derivatives with respect to the integer variables. Extensive numerical tests show, that MIQPSOA significantly improves the reliability of well-known linear outer approximation methods on the one hand, while it also improves the efficiency in terms of the number of function evaluations.

In the second part of this thesis, we review and develop theory for solving strictly convex mixed-integer quadratic programming problems. Inspired by powerful state-of-the-art MILP solvers, the main focus is the application of warmstarts and the generation of cutting planes. The construction of cutting planes for MIQPs implicate some major difficulties compared to the well-known case of MILP. It turns out, that the proposed cut generation procedure for disjunctive cutting planes is very efficient. However, if cutting planes exist, they can significantly reduce the computation time.

There are some closely related topics that need to be investigated in the future. First, we want to propose a new solver for convex MINLP problems that purely relies on MIQP subproblems. It seems to be possible to prove convergence of such a sequential mixed-integer quadratic programming algorithm by exploiting the results of Leyffer [76] as pointed out in Section 3.3. Moreover, we need to prove, that MINLP algorithms can guarantee global optimality for convex MINLP problems, if exact gradients are not available, e.g., for setting up the linear outer approximation master problem. Exploiting the theory of disjunctive programming, one might be able to show, that descent directions approximated at neighbored grid-points, i.e., numerical forward or backward differences with step-size one, are sufficient.

The numerical results of the MIQP test cases library show, that a significant speed-up is still required in order to handle larger MINLP instances by MIQP-based solution

techniques. Therefore, one main task for future research is to further improve the performance of MIQL. There are plenty of ways to continue the development of the mixed-integer quadratic programming solver. Since presolving techniques seem to have little success, we propose to concentrate on advanced branching rules and an improved cut generation management. The cut generation process needs to be extended, such that cut generation is not only applied at the root node but also on lower levels of the branch-and-bound tree. Such strategy seems to be profitable, since the implemented cut generator is sufficiently fast and the probability of the existence of cutting planes is increased, if the number of constraints grows.

APPENDIX

A. DETAILED MINLP RESULTS

In the sequel we present the detailed results of all MINLP solvers for both the 100 academic test cases and the test set of 55 instances from petroleum industry provided by our cooperation partner Shell. For all solvers and all test cases we consider the following criteria.

Criterion	Description
TP	Index of the test problem
NAME	Name of the test problem, see also Table A.3
IFAIL	Termination flag indicating successful or false termination
N_f	Total number of function evaluations including calls to approximate derivatives
F	Objective value at the solution
OBJ_ERR	Deviation from the best-known solution
VIOL	Maximal constraint violation at solution
TIME	Running time in seconds.

Tab. A.1: Criteria for detailed MINLP Results

A.1 Academic Test Set

We considered a test set containing 100 academic test cases, which is published by Schittkowski [95]. First we describe the characteristics of the test cases. The described parameter are explained in Table A.2.

Criterion	Description
TP	Index of the test problem
NAME	Name of the test problem, see also Table A.3
REF	Reference of the test problem
n_c	Number of continuous variables
n_i	Number of integer variables
n_b	Number of binary variables
m_e	Number of equality constraints
m	Total number of constraints
f^*	Best-known objective value

Tab. A.2: Description of Academic Test Cases

The subsequent table presents the problems with a reference, the best-known objective value and the characteristics of the problem.

TP	NAME	REF	n_c	n_i	n_b	m_e	m	f^*
1	MITP1		2	3	0	0	1	-0.10010E+05
2	QIP1		0	4	0	0	4	-0.20000E+02
3	MITP2		2	0	3	0	7	0.35000E+01
4	ASAADI11	[5]	1	3	0	0	3	-0.40957E+02
5	ASAADI12	[5]	0	4	0	0	3	-0.38000E+02
6	ASAADI21	[5]	3	4	0	0	4	0.69490E+03
7	ASAADI22	[5]	0	7	0	0	4	0.70000E+03
8	ASAADI31	[5]	4	6	0	0	8	0.37220E+02
9	ASAADI32	[5]	0	10	0	0	8	0.43000E+02
10	DIRTY		12	13	0	0	10	-0.30472E+01
11	BRAAK1	[30]	4	3	0	0	2	0.10000E+01
12	BRAAK2	[30]	4	3	0	0	4	-0.27183E+01
13	BRAAK3	[30]	4	3	0	0	4	-0.19656E+00
14	DEX2	[36]	0	2	0	0	2	-0.56938E+02
15	TP83	[65]	3	2	0	0	6	-0.30666E+05
16	WP02	[109]	1	1	0	0	2	-0.24444E+01
17	NVS01	[34]	1	2	0	1	3	0.12470E+02
18	NVS02	[34]	3	5	0	3	3	0.59642E+01
19	NVS03	[34]	0	2	0	0	2	0.16000E+02
20	NVS04	[34]	0	2	0	0	0	0.72000E+00
21	NVS05	[34]	6	2	0	4	9	0.54709E+01
22	NVS06	[34]	0	2	0	0	0	0.17703E+01
23	NVS07	[34]	0	3	0	0	2	0.40000E+01
24	NVS08	[34]	1	2	0	0	3	0.23450E+02
25	NVS09	[34]	0	10	0	0	0	-0.43134E+02
26	NVS10	[34]	0	2	0	0	2	-0.31080E+03
27	NVS11	[34]	0	3	0	0	2	-0.43100E+03

TP	NAME	REF	n_c	n_i	n_b	m_e	m	f^*
28	NVS12	[34]	0	4	0	0	4	-0.48120E+03
29	NVS13	[34]	0	5	0	0	5	-0.58520E+03
30	NVS14	[34]	3	5	0	3	3	-0.40358E+05
31	NVS15	[34]	0	3	0	0	1	0.10000E+01
32	NVS16	[34]	0	2	0	0	0	0.70312E+00
33	NVS17	[34]	0	7	0	0	7	-0.11004E+04
34	NVS18	[34]	0	6	0	0	6	-0.77840E+03
35	NVS19	[34]	0	8	0	0	8	-0.10984E+04
36	NVS20	[34]	11	5	0	0	8	0.23092E+03
37	NVS21	[34]	1	2	0	0	2	-0.56848E+01
38	NVS22	[34]	4	4	0	4	9	0.60582E+01
39	NVS23	[34]	0	9	0	0	9	-0.11252E+04
40	NVS24	[34]	0	10	0	0	10	-0.10332E+04
41	GEAR	[34]	0	4	0	0	0	0.10000E+01
42	GEAR2	[34]	4	24	0	4	4	0.10000E+01
43	GEAR3	[34]	4	4	0	4	4	0.10000E+01
44	GEAR4	[34]	2	4	0	1	1	0.16434E+01
45	WINDFAC	[34]	11	3	0	13	13	0.25449E+00
46	DG1	[43]	3	0	3	0	6	0.60097E+01
47	DG2	[43]	6	0	5	0	14	0.17214E+02
48	DG3	[43]	9	0	8	2	23	0.68010E+02
49	FLOUDAS1	[53]	2	0	3	2	5	0.76672E+01
50	FLOUDAS2	[53]	2	0	1	0	3	0.10765E+01
51	FLOUDAS3	[53]	3	0	4	0	9	0.45796E+01
52	FLOUDAS4	[53]	3	0	8	3	7	-0.94347E+00
53	FLOUDAS5	[53]	0	2	0	0	4	0.31000E+02
54	FLOUDAS6	[53]	1	1	0	0	3	-0.17000E+02
55	OAER	[34]	6	0	3	3	7	-0.19231E+01
56	SPRING	[34]	5	1	11	5	8	0.84625E+00
57	DAKOTA	[44]	2	2	0	0	2	0.13634E+01
58	PROB02	[34]	0	6	0	0	8	0.11224E+06
59	PROB03	[34]	0	2	0	0	1	0.10000E+02
60	PROB10	[34]	1	1	0	0	2	0.34455E+01
61	BATCH	[34]	23	0	24	12	73	0.28551E+00
62	BATCHDES	[34]	10	0	9	6	19	0.16743E+06
63	DU_OPT5	[34]	7	13	0	0	9	0.80737E+01
64	DU_OPT	[34]	7	13	0	0	9	0.35563E+01
65	ST_E13	[34]	1	0	1	0	2	0.20000E+01
66	ST_E32	[34]	16	19	0	17	18	-0.14304E+01
67	ST_E36	[34]	1	1	0	1	2	-0.24600E+03
68	ST_E38	[34]	2	2	0	0	3	0.71977E+04
69	ST_MIQP1	[34]	0	0	5	0	1	0.28100E+03
70	ST_MIQP2	[34]	0	4	0	0	3	0.20000E+01
71	ST_MIQP3	[34]	0	2	0	0	1	-0.60000E+01
72	ST_MIQP4	[34]	3	0	3	0	4	-0.45740E+04
73	ST_MIQP5	[34]	5	2	0	0	13	-0.33389E+03
74	ST_TEST1	[34]	0	5	0	0	1	0.10000E+01

TP	NAME	REF	n_c	n_i	n_b	m_e	m	f^*
75	ST_TEST2	[34]	0	6	0	0	2	-0.92500E+01
76	ST_TEST3	[34]	0	13	0	0	10	-0.70000E+01
77	ST_TEST4	[34]	0	6	0	0	5	-0.70000E+01
78	ST_TEST5	[34]	0	10	0	0	11	-0.11000E+03
79	ST_TEST6	[34]	0	0	10	0	5	0.47100E+03
80	ST_TEST8	[34]	0	24	0	0	20	-0.29605E+05
81	ST_TESTGR1	[34]	0	10	0	0	5	-0.12812E+02
82	ST_TESTGR3	[34]	0	20	0	0	20	-0.20590E+02
83	ST_TESTPH4	[34]	0	3	0	0	10	-0.80500E+02
84	TLN2	[34]	0	6	2	0	12	0.53000E+01
85	TLN4	[34]	0	20	4	0	24	0.83000E+01
86	TLN5	[34]	0	30	5	0	30	0.10300E+02
87	TLN6	[34]	0	42	6	0	36	0.15300E+02
88	PROCSEL	[34]	7	0	3	4	7	-0.19231E+01
89	TLOSS	[34]	0	42	6	0	53	0.16300E+02
90	TLTR	[34]	0	36	12	0	54	0.48067E+02
91	ALAN	[34]	4	0	4	2	7	0.28990E+01
92	MEANVARX	[34]	21	0	14	8	44	0.14369E+02
93	HMITTELMANN	[34]	0	0	16	0	7	0.13000E+02
94	MIP_EX	[61]	2	0	3	0	7	0.35000E+01
95	MGRID_CYCLES1	[100]	0	5	0	0	1	0.80000E+01
96	MGRID_CYCLES2	[100]	0	10	0	0	1	0.30000E+03
97	CROP5	[98]	0	5	0	0	3	0.10041E+00
98	CROP20	[98]	0	20	0	0	3	0.13178E+00
99	CROP50	[98]	0	50	0	0	3	0.37459E+00
100	CROP100	[98]	0	100	0	0	3	0.15684E+01

Tab. A.3: Characteristics of Academic Test Cases

In the remainder of this section we present the detailed numerical results for all solver on the 100 academic test problems. The results of the mixed-integer sequential quadratic programming method MISQP are reviewd in Table A.4.

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	MITP1	0	281	-0.100097E+05	0.00E+00	0.00E+00	0.00
2	QIP1	0	22	-0.200000E+02	0.00E+00	0.00E+00	0.00
3	MITP2	0	48	0.350000E+01	-0.95E-10	0.12E-09	0.00
4	ASAADI11	0	115	-0.409574E+02	-0.20E-04	0.47E-10	0.00
5	ASAADI12	0	144	-0.380000E+02	0.00E+00	0.00E+00	0.00
6	ASAADI21	0	338	0.694903E+03	0.48E-08	0.00E+00	0.00
7	ASAADI22	0	721	0.700000E+03	0.00E+00	0.00E+00	0.03
8	ASAADI31	0	332	0.372195E+02	-0.18E-05	0.63E-12	0.01
9	ASAADI32	0	202	0.430000E+02	0.00E+00	0.00E+00	0.02
10	DIRTY	0	5416	-0.304659E+01	0.21E-03	0.00E+00	0.39
11	BRAAK1	0	707	0.100424E+01	0.42E-02	0.00E+00	0.01
12	BRAAK2	0	1217	-0.271813E+01	0.55E-04	0.00E+00	0.02
13	BRAAK3	0	429	-0.196559E+00	0.66E-05	0.00E+00	0.01
14	DEX2	0	33	-0.569375E+02	0.00E+00	0.00E+00	0.00
15	TP83	0	126	-0.306655E+05	0.58E-07	0.25E-07	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
16	WP02	0	35	-0.244444E+01	-0.15E-05	0.00E+00	0.00
17	NVS01	0	108	0.124697E+02	-0.95E-07	0.59E-14	0.00
18	NVS02	0	363	0.596418E+01	-0.80E-07	0.27E-12	0.01
19	NVS03	0	37	0.160000E+02	0.00E+00	0.00E+00	0.00
20	NVS04	0	31	0.720000E+00	0.83E-14	0.00E+00	0.00
21	NVS05	0	635	0.547093E+01	-0.14E-06	0.24E-06	0.02
22	NVS06	0	64	0.177031E+01	0.28E-06	0.00E+00	0.00
23	NVS07	0	22	0.400000E+01	0.00E+00	0.00E+00	0.00
24	NVS08	0	235	0.234497E+02	-0.13E-06	0.12E-06	0.00
25	NVS09	0	32	-0.431343E+02	0.71E-07	0.00E+00	0.00
26	NVS10	0	37	-0.310800E+03	-0.18E-15	0.00E+00	0.00
27	NVS11	0	223	-0.431000E+03	0.00E+00	0.00E+00	0.00
28	NVS12	0	99	-0.481200E+03	0.00E+00	0.00E+00	0.00
29	NVS13	0	290	-0.585200E+03	0.00E+00	0.00E+00	0.01
30	NVS14	0	280	-0.403582E+05	-0.12E-06	0.43E-13	0.01
31	NVS15	0	33	0.100000E+01	0.00E+00	0.00E+00	0.00
32	NVS16	0	28	0.703125E+00	0.00E+00	0.00E+00	0.00
33	NVS17	0	247	-0.110040E+04	-0.21E-15	0.00E+00	0.01
34	NVS18	0	601	-0.778400E+03	0.00E+00	0.00E+00	0.02
35	NVS19	0	817	-0.109840E+04	0.00E+00	0.00E+00	0.04
36	NVS20	0	730	0.230973E+03	0.22E-03	0.15E-09	0.03
37	NVS21	0	164	-0.568478E+01	0.88E-07	0.77E-10	0.00
38	NVS22	0	209	0.605822E+01	0.00E+00	0.38E-12	0.01
39	NVS23	0	946	-0.112520E+04	0.00E+00	0.00E+00	0.06
40	NVS24	0	740	-0.102380E+04	0.91E-02	0.00E+00	0.04
41	GEAR	0	275	0.100000E+01	0.45E-07	0.00E+00	0.00
42	GEAR2	0	841	0.100000E+01	0.41E-06	0.26E-11	2.03
43	GEAR3	0	912	0.100000E+01	0.16E-07	0.75E-10	0.16
44	GEAR4	4	1374	0.000000E+00	-0.10E+01	0.13E-04	1.64
45	WINDFAC	0	964	0.254487E+00	0.21E-08	0.11E-08	0.06
46	DG1	0	205	0.600876E+01	-0.16E-03	0.92E-08	0.00
47	DG2	0	790	0.172142E+02	0.27E-05	0.00E+00	0.05
48	DG3	0	938	0.680097E+02	-0.38E-05	0.60E-12	0.07
49	FLOUDAS1	0	24	0.766718E+01	-0.33E-08	0.33E-08	0.00
50	FLOUDAS2	0	28	0.107654E+01	0.29E-05	0.55E-12	0.00
51	FLOUDAS3	0	253	0.457958E+01	-0.98E-07	0.23E-06	0.01
52	FLOUDAS4	0	445	-0.943471E+00	-0.17E-10	0.11E-09	0.04
53	FLOUDAS5	0	17	0.310000E+02	0.00E+00	0.00E+00	0.00
54	FLOUDAS6	0	17	-0.170000E+02	-0.66E-12	0.23E-11	0.00
55	OAER	0	160	-0.192310E+01	0.25E-06	0.47E-09	0.00
56	SPRING	0	1316	0.846246E+00	-0.40E-07	0.13E-09	0.14
57	DAKOTA	0	96	0.136340E+01	-0.28E-12	0.31E-13	0.00
58	PROB02	0	130	0.112235E+06	0.00E+00	0.00E+00	0.00
59	PROB03	0	15	0.100000E+02	0.00E+00	0.00E+00	0.00
60	PROB10	0	16	0.344550E+01	-0.17E-10	0.19E-10	0.00
61	BATCH	0	6303	0.285506E+00	0.16E-05	0.36E-07	8.93
62	BATCHDES	0	600	0.239960E+06	0.43E+00	0.10E-10	0.02

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
63	DU_OPT5	0	2218	0.202563E+02	0.15E+01	0.00E+00	0.10
64	DU_OPT	0	3196	0.684523E+01	0.92E+00	0.00E+00	0.15
65	ST_E13	0	12	0.223607E+01	0.12E+00	0.00E+00	0.00
66	ST_E32	0	631	-0.143041E+01	0.12E-07	0.35E-11	0.22
67	ST_E36	0	245	-0.168310E+03	0.32E+00	0.52E-06	0.00
68	ST_E38	0	389	0.719773E+04	0.61E-12	0.20E-12	0.01
69	ST_MIQP1	0	36	0.281000E+03	0.00E+00	0.00E+00	0.00
70	ST_MIQP2	0	65	0.200000E+01	0.00E+00	0.00E+00	0.00
71	ST_MIQP3	0	13	-0.600000E+01	0.00E+00	0.00E+00	0.00
72	ST_MIQP4	0	28	-0.457400E+04	-0.16E-09	0.30E-08	0.00
73	ST_MIQP5	0	123	-0.333889E+03	0.33E-07	0.83E-10	0.00
74	ST_TEST1	0	6	0.100000E+01	0.00E+00	0.00E+00	0.00
75	ST_TEST2	0	42	-0.925000E+01	0.00E+00	0.00E+00	0.00
76	ST_TEST3	0	59	-0.700000E+01	0.00E+00	0.00E+00	0.00
77	ST_TEST4	0	45	-0.700000E+01	0.00E+00	0.00E+00	0.00
78	ST_TEST5	0	44	-0.110000E+03	0.00E+00	0.00E+00	0.00
79	ST_TEST6	0	77	0.471000E+03	0.00E+00	0.00E+00	0.01
80	ST_TEST8	0	494	-0.296050E+05	0.00E+00	0.00E+00	0.10
81	ST_TESTGR1	0	134	-0.127976E+02	0.11E-02	0.00E+00	0.01
82	ST_TESTGR3	0	483	-0.205900E+02	0.00E+00	0.00E+00	0.28
83	ST_TESTPH4	0	24	-0.805000E+02	0.00E+00	0.00E+00	0.00
84	TLN2	0	106	0.530000E+01	0.00E+00	0.00E+00	0.02
85	TLN4	0	1492	0.850000E+01	0.24E-01	0.00E+00	7.73
86	TLN5	0	1708	0.106000E+02	0.29E-01	0.00E+00	9.81
87	TLN6	0	2243	0.163000E+02	0.65E-01	0.00E+00	19.37
88	PROCSEL	0	353	-0.192310E+01	0.14E-06	0.76E-11	0.01
89	TLOSS	0	1260	0.163000E+02	0.00E+00	0.00E+00	11.57
90	TLTR	0	305	0.480667E+02	-0.74E-15	0.00E+00	0.72
91	ALAN	0	477	0.292500E+01	0.90E-02	0.10E-07	0.02
92	MEANVARX	0	2024	0.141897E+02	-0.12E-01	0.23E-11	0.61
93	HMITTELMANN	0	51	0.160000E+02	0.23E+00	0.00E+00	0.02
94	MIP_EX	0	36	0.350000E+01	0.00E+00	0.00E+00	0.00
95	MGRID_CYCLES1	0	79	0.800000E+01	0.00E+00	0.00E+00	0.00
96	MGRID_CYCLES2	0	843	0.300000E+03	0.00E+00	0.00E+00	0.04
97	CROP5	0	66	0.100409E+00	-0.51E-05	0.00E+00	0.00
98	CROP20	0	5228	0.144696E+00	0.98E-01	0.00E+00	6.87
99	CROP50	0	1774	0.374595E+00	0.13E-04	0.00E+00	11.10
100	CROP100	0	38466	0.455721E-02	-0.10E+01	0.00E+00	155.45

Tab. A.4: Detailed Results of MISQP for the Academic Test Set

The results of the basic version of the mixed-integer sequential quadratic programming method MISQP without fine-tuning are reviewed in Table A.5.

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	MITP1	0	197	-0.100097E+05	0.00E+00	0.00E+00	0.00
2	QIP1	0	28	-0.200000E+02	0.00E+00	0.00E+00	0.00
3	MITP2	0	52	0.500000E+01	0.43E+00	0.66E-11	0.00
4	ASAADI11	0	63	-0.256985E+02	0.37E+00	0.21E-08	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
5	ASAADI12	0	78	-0.380000E+02	0.00E+00	0.00E+00	0.00
6	ASAADI21	0	295	0.694904E+03	0.17E-05	0.00E+00	0.00
7	ASAADI22	0	97	0.714000E+03	0.20E-01	0.00E+00	0.00
8	ASAADI31	0	414	0.114042E+03	0.21E+01	0.28E-08	0.01
9	ASAADI32	0	376	0.520000E+02	0.21E+00	0.00E+00	0.01
10	DIRTY	0	1285	-0.304410E+01	0.10E-02	0.55E-11	0.05
11	BRAAK1	0	333	0.113487E+01	0.13E+00	0.00E+00	0.00
12	BRAAK2	0	113	-0.999739E+00	0.63E+00	0.00E+00	0.00
13	BRAAK3	0	294	-0.321242E-01	0.84E+00	0.00E+00	0.00
14	DEX2	0	36	-0.569375E+02	0.00E+00	0.00E+00	0.00
15	TP83	0	92	-0.302128E+05	0.15E-01	0.17E-11	0.00
16	WP02	0	27	-0.244444E+01	-0.14E-05	0.00E+00	0.00
17	NVS01	0	54	0.144199E+02	0.16E+00	0.47E-13	0.00
18	NVS02	0	84	0.725016E+01	0.22E+00	0.19E-12	0.00
19	NVS03	0	27	0.160000E+02	0.00E+00	0.00E+00	0.00
20	NVS04	0	15	0.720000E+00	0.83E-14	0.00E+00	0.00
21	NVS05	0	99	0.125909E+02	0.13E+01	0.90E-07	0.00
22	NVS06	0	8	0.177031E+01	0.28E-06	0.00E+00	0.00
23	NVS07	0	12	0.400000E+01	0.00E+00	0.00E+00	0.00
24	NVS08	0	69	0.260000E+02	0.11E+00	0.11E-10	0.00
25	NVS09	0	642	-0.431343E+02	0.71E-07	0.00E+00	0.01
26	NVS10	0	29	-0.310800E+03	-0.18E-15	0.00E+00	0.00
27	NVS11	0	54	-0.431000E+03	0.00E+00	0.00E+00	0.00
28	NVS12	0	60	-0.481200E+03	0.00E+00	0.00E+00	0.00
29	NVS13	0	135	-0.585200E+03	0.00E+00	0.00E+00	0.00
30	NVS14	0	42	-0.386352E+05	0.43E-01	0.40E-12	0.00
31	NVS15	0	39	0.100000E+01	0.00E+00	0.00E+00	0.00
32	NVS16	0	12	0.703125E+00	0.00E+00	0.00E+00	0.00
33	NVS17	0	79	-0.110040E+04	-0.21E-15	0.00E+00	0.00
34	NVS18	0	138	-0.776600E+03	0.23E-02	0.00E+00	0.00
35	NVS19	0	136	-0.108920E+04	0.84E-02	0.00E+00	0.00
36	NVS20	0	472	0.313343E+03	0.36E+00	0.30E-11	0.01
37	NVS21	0	69	-0.545739E+01	0.40E-01	0.55E-08	0.00
38	NVS22	0	120	0.822135E+01	0.36E+00	0.12E-11	0.00
39	NVS23	0	349	-0.112520E+04	0.00E+00	0.00E+00	0.02
40	NVS24	0	389	-0.102380E+04	0.91E-02	0.00E+00	0.02
41	GEAR	0	117	0.100000E+01	0.50E-06	0.00E+00	0.00
42	GEAR2	0	58	0.117491E+01	0.17E+00	0.25E-10	0.01
43	GEAR3	0	9	0.173226E+01	0.73E+00	0.00E+00	0.00
44	GEAR4	4	280	0.100000E+02	0.51E+01	0.18E-04	0.12
45	WINDFAC	0	239	0.750000E+00	0.19E+01	0.58E-12	0.01
46	DG1	0	106	0.600876E+01	-0.16E-03	0.92E-08	0.00
47	DG2	0	132	0.172142E+02	0.27E-05	0.15E-10	0.00
48	DG3	0	198	0.680097E+02	-0.38E-05	0.50E-11	0.01
49	FLOUDAS1	0	24	0.766718E+01	-0.33E-08	0.16E-08	0.00
50	FLOUDAS2	0	8	0.125000E+01	0.16E+00	0.24E-11	0.00
51	FLOUDAS3	0	16	0.580685E+01	0.27E+00	0.37E-10	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
52	FLOUDAS4	0	206	-0.601600E+00	0.36E+00	0.30E-10	0.01
53	FLOUDAS5	0	5	0.340000E+02	0.97E-01	0.00E+00	0.00
54	FLOUDAS6	0	17	-0.170000E+02	-0.66E-12	0.23E-11	0.00
55	OAER	0	70	-0.192310E+01	0.25E-06	0.68E-09	0.00
56	SPRING	0	307	0.128990E+01	0.52E+00	0.69E-07	0.01
57	DAKOTA	0	54	0.136340E+01	-0.25E-06	0.23E-07	0.00
58	PROB02	0	65	0.152255E+06	0.36E+00	0.00E+00	0.00
59	PROB03	0	5	0.100000E+02	0.00E+00	0.00E+00	0.00
60	PROB10	0	8	0.344550E+01	-0.47E-11	0.43E-11	0.00
61	BATCH	0	3416	0.470189E+00	0.65E+00	0.99E-06	1.01
62	BATCHDES	0	360	0.267226E+06	0.60E+00	0.64E-08	0.01
63	DU_OPT5	0	2059	0.679541E+03	0.83E+02	0.00E+00	0.09
64	DU_OPT	0	1320	0.304440E+04	0.86E+03	0.00E+00	0.05
65	ST_E13	0	12	0.223607E+01	0.12E+00	0.00E+00	0.00
66	ST_E32	0	445	-0.701910E+00	0.51E+00	0.30E-10	0.09
67	ST_E36	0	213	-0.168310E+03	0.32E+00	0.52E-06	0.00
68	ST_E38	0	145	0.744036E+04	0.34E-01	0.00E+00	0.00
69	ST_MIQP1	0	12	0.281000E+03	0.00E+00	0.00E+00	0.00
70	ST_MIQP2	0	24	0.500000E+01	0.15E+01	0.00E+00	0.00
71	ST_MIQP3	0	3	0.000000E+00	0.10E+01	0.00E+00	0.00
72	ST_MIQP4	0	14	-0.457400E+04	-0.16E-09	0.30E-08	0.00
73	ST_MIQP5	0	81	-0.239889E+03	0.28E+00	0.79E-11	0.00
74	ST_TEST1	0	6	0.100000E+01	0.00E+00	0.00E+00	0.00
75	ST_TEST2	0	14	-0.925000E+01	0.00E+00	0.00E+00	0.00
76	ST_TEST3	0	29	-0.600000E+01	0.14E+00	0.00E+00	0.00
77	ST_TEST4	0	20	-0.600000E+01	0.14E+00	0.00E+00	0.00
78	ST_TEST5	0	22	-0.110000E+03	0.00E+00	0.00E+00	0.00
79	ST_TEST6	0	22	0.471000E+03	0.00E+00	0.00E+00	0.00
80	ST_TEST8	0	425	-0.296050E+05	0.00E+00	0.00E+00	0.10
81	ST_TESTGR1	0	360	-0.122298E+02	0.45E-01	0.00E+00	0.01
82	ST_TESTGR3	0	1138	-0.178159E+02	0.13E+00	0.00E+00	0.23
83	ST_TESTPH4	0	13	-0.805000E+02	0.00E+00	0.00E+00	0.00
84	TLN2	0	41	0.530000E+01	0.00E+00	0.00E+00	0.00
85	TLN4	0	168	0.111000E+02	0.34E+00	0.00E+00	0.19
86	TLN5	0	245	0.126000E+02	0.22E+00	0.00E+00	0.71
87	TLN6	0	830	0.206000E+02	0.35E+00	0.00E+00	2.98
88	PROCSEL	0	89	-0.141100E+01	0.27E+00	0.89E-06	0.00
89	TLOSS	0	259	0.181000E+02	0.11E+00	0.00E+00	0.55
90	TLTR	0	197	0.480667E+02	-0.74E-15	0.00E+00	0.61
91	ALAN	0	36	0.312353E+01	0.77E-01	0.81E-10	0.00
92	MEANVARX	0	324	0.141897E+02	-0.12E-01	0.23E-11	0.05
93	HMITTELMANN	0	34	0.160000E+02	0.23E+00	0.00E+00	0.01
94	MIP_EX	0	12	0.350000E+01	0.00E+00	0.00E+00	0.00
95	MGRID_CYCLES1	0	45	0.800000E+01	0.00E+00	0.00E+00	0.00
96	MGRID_CYCLES2	0	700	0.300000E+03	0.00E+00	0.00E+00	0.03
97	CROP5	0	71	0.100409E+00	-0.51E-05	0.00E+00	0.00
98	CROP20	0	821	0.131785E+00	-0.17E-07	0.00E+00	0.05

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
99	CROP50	0	4802	0.374595E+00	0.13E-04	0.00E+00	1.22
100	CROP100	0	18913	0.196171E-01	-0.99E+00	0.00E+00	24.64

Tab. A.5: Detailed Results of MISQP without Fine-tuning for the Academic Test Set

The detailed results of the code MISQPOA are presented in Table A.6.

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	MITP1	0	301	-0.100097E+05	0.00E+00	0.00E+00	0.00
2	QIP1	0	35	-0.200000E+02	0.00E+00	0.00E+00	0.00
3	MITP2	0	144	0.350000E+01	-0.95E-10	0.12E-09	0.00
4	ASAADI11	0	254	-0.409574E+02	-0.20E-04	0.47E-10	0.00
5	ASAADI12	0	159	-0.380000E+02	0.00E+00	0.00E+00	0.00
6	ASAADI21	0	648	0.694903E+03	0.48E-08	0.00E+00	0.01
7	ASAADI22	0	3978	0.700000E+03	0.00E+00	0.00E+00	0.27
8	ASAADI31	0	1321	0.372195E+02	-0.18E-05	0.63E-12	0.03
9	ASAADI32	0	2518	0.430000E+02	0.00E+00	0.00E+00	0.45
10	DIRTY	0	9351	-0.304695E+01	0.97E-04	0.21E-10	0.66
11	BRAAK1	0	2025	0.100424E+01	0.42E-02	0.00E+00	0.03
12	BRAAK2	0	1946	-0.271828E+01	-0.27E-06	0.00E+00	0.03
13	BRAAK3	0	452	-0.196559E+00	0.66E-05	0.00E+00	0.00
14	DEX2	0	42	-0.569375E+02	0.00E+00	0.00E+00	0.00
15	TP83	0	216	-0.306655E+05	0.58E-07	0.25E-07	0.00
16	WP02	0	155	-0.244444E+01	-0.18E-05	0.00E+00	0.00
17	NVS01	0	122	0.124697E+02	-0.95E-07	0.59E-14	0.00
18	NVS02	0	705	0.596418E+01	-0.80E-07	0.27E-12	0.02
19	NVS03	0	46	0.160000E+02	0.00E+00	0.00E+00	0.00
20	NVS04	0	183	0.720000E+00	0.83E-14	0.00E+00	0.00
21	NVS05	0	1754	0.547093E+01	-0.14E-06	0.24E-06	0.04
22	NVS06	0	73	0.177031E+01	0.28E-06	0.00E+00	0.00
23	NVS07	0	403	0.400000E+01	0.00E+00	0.00E+00	0.02
24	NVS08	0	446	0.234497E+02	-0.13E-06	0.12E-06	0.00
25	NVS09	0	63	-0.431343E+02	0.71E-07	0.00E+00	0.00
26	NVS10	0	46	-0.310800E+03	-0.18E-15	0.00E+00	0.00
27	NVS11	0	236	-0.431000E+03	0.00E+00	0.00E+00	0.00
28	NVS12	0	116	-0.481200E+03	0.00E+00	0.00E+00	0.00
29	NVS13	0	311	-0.585200E+03	0.00E+00	0.00E+00	0.01
30	NVS14	0	312	-0.403582E+05	-0.12E-06	0.43E-13	0.01
31	NVS15	0	247	0.100000E+01	0.00E+00	0.00E+00	0.00
32	NVS16	0	43	0.703125E+00	0.00E+00	0.00E+00	0.00
33	NVS17	0	276	-0.110040E+04	-0.21E-15	0.00E+00	0.01
34	NVS18	0	626	-0.778400E+03	0.00E+00	0.00E+00	0.02
35	NVS19	0	850	-0.109840E+04	0.00E+00	0.00E+00	0.04
36	NVS20	0	1801	0.230973E+03	0.22E-03	0.15E-09	0.04
37	NVS21	0	286	-0.568478E+01	0.88E-07	0.77E-10	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
38	NVS22	0	603	0.605822E+01	0.00E+00	0.38E-12	0.01
39	NVS23	0	983	-0.112520E+04	0.00E+00	0.00E+00	0.07
40	NVS24	0	778	-0.102380E+04	0.91E-02	0.00E+00	0.04
41	GEAR	0	832	0.100000E+01	0.18E-07	0.00E+00	0.01
42	GEAR2	0	3052	0.100000E+01	0.41E-06	0.26E-11	7.58
43	GEAR3	0	1919	0.100000E+01	0.24E-08	0.12E-09	0.17
44	GEAR4	4	3904	0.000000E+00	-0.10E+01	0.10E+01	2.88
45	WINDFAC	0	4757	0.254487E+00	-0.65E-08	0.50E-12	0.22
46	DG1	0	585	0.600876E+01	-0.16E-03	0.92E-08	0.01
47	DG2	0	1098	0.172142E+02	0.27E-05	0.15E-10	0.04
48	DG3	0	1633	0.680097E+02	-0.38E-05	0.50E-11	0.11
49	FLOUDAS1	0	139	0.766718E+01	-0.33E-08	0.16E-08	0.00
50	FLOUDAS2	0	65	0.107654E+01	0.29E-05	0.55E-12	0.00
51	FLOUDAS3	0	896	0.457958E+01	-0.98E-07	0.17E-06	0.03
52	FLOUDAS4	0	753	-0.943471E+00	-0.39E-10	0.46E-08	0.07
53	FLOUDAS5	0	92	0.310000E+02	0.00E+00	0.00E+00	0.00
54	FLOUDAS6	0	25	-0.170000E+02	-0.66E-12	0.23E-11	0.00
55	OAER	0	196	-0.192310E+01	0.25E-06	0.68E-09	0.00
56	SPRING	0	2099	0.846246E+00	-0.40E-07	0.13E-09	0.20
57	DAKOTA	0	112	0.136340E+01	-0.28E-12	0.31E-13	0.00
58	PROB02	0	155	0.112235E+06	0.00E+00	0.00E+00	0.00
59	PROB03	0	24	0.100000E+02	0.00E+00	0.00E+00	0.00
60	PROB10	0	73	0.344550E+01	-0.17E-10	0.19E-10	0.00
61	BATCH	0	10635	0.285506E+00	0.17E-05	0.20E-07	15.11
62	BATCHDES	0	658	0.239960E+06	0.43E+00	0.10E-10	0.02
63	DU_OPT5	0	7047	0.202563E+02	0.15E+01	0.00E+00	0.79
64	DU_OPT	0	5924	0.684523E+01	0.92E+00	0.00E+00	0.28
65	ST_E13	0	44	0.200000E+01	0.47E-08	0.00E+00	0.00
66	ST_E32	4	2750	-0.184000E+02	-0.12E+02	0.32E+03	0.59
67	ST_E36	0	391	-0.246000E+03	-0.18E-10	0.14E-09	0.00
68	ST_E38	0	662	0.719773E+04	0.61E-12	0.20E-12	0.01
69	ST_MIQP1	0	116	0.281000E+03	0.00E+00	0.00E+00	0.00
70	ST_MIQP2	0	169	0.200000E+01	0.00E+00	0.00E+00	0.00
71	ST_MIQP3	0	22	-0.600000E+01	0.00E+00	0.00E+00	0.00
72	ST_MIQP4	0	48	-0.457400E+04	-0.16E-09	0.30E-08	0.00
73	ST_MIQP5	0	145	-0.333889E+03	0.33E-07	0.83E-10	0.00
74	ST_TEST1	0	50	0.100000E+01	0.00E+00	0.00E+00	0.00
75	ST_TEST2	0	61	-0.925000E+01	0.00E+00	0.00E+00	0.00
76	ST_TEST3	0	100	-0.700000E+01	0.00E+00	0.00E+00	0.00
77	ST_TEST4	0	65	-0.700000E+01	0.00E+00	0.00E+00	0.00
78	ST_TEST5	0	75	-0.110000E+03	0.00E+00	0.00E+00	0.00
79	ST_TEST6	0	312	0.471000E+03	0.00E+00	0.00E+00	0.02
80	ST_TEST8	0	577	-0.296050E+05	0.00E+00	0.00E+00	0.10
81	ST_TESTGR1	0	170	-0.127976E+02	0.11E-02	0.00E+00	0.01
82	ST_TESTGR3	0	326	-0.205900E+02	0.00E+00	0.00E+00	0.17
83	ST_TESTPH4	0	36	-0.805000E+02	0.00E+00	0.00E+00	0.00
84	TLN2	0	302	0.530000E+01	0.00E+00	0.00E+00	0.03

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
85	TLN4	0	3962	0.830000E+01	0.00E+00	0.00E+00	9.84
86	TLN5	0	6872	0.106000E+02	0.29E-01	0.00E+00	25.55
87	TLN6	0	7954	0.163000E+02	0.65E-01	0.00E+00	50.10
88	PROCSEL	0	658	-0.192310E+01	0.14E-06	0.76E-11	0.02
89	TLOSS	0	2121	0.163000E+02	0.00E+00	0.00E+00	13.49
90	TLTR	0	1491	0.611333E+02	0.27E+00	0.00E+00	6.61
91	ALAN	0	758	0.292500E+01	0.90E-02	0.10E-07	0.03
92	MEANVARX	0	3991	0.141897E+02	-0.12E-01	0.23E-11	1.21
93	HMITTELMANN	0	167	0.160000E+02	0.23E+00	0.00E+00	0.04
94	MIP_EX	0	138	0.350000E+01	0.00E+00	0.00E+00	0.00
95	MGRID_CYCLES1	0	219	0.800000E+01	0.00E+00	0.00E+00	0.02
96	MGRID_CYCLES2	0	1566	0.300000E+03	0.00E+00	0.00E+00	0.10
97	CROP5	0	87	0.100409E+00	-0.51E-05	0.00E+00	0.00
98	CROP20	0	5451	0.148424E+00	0.13E+00	0.00E+00	6.51
99	CROP50	0	7841	0.374595E+00	0.13E-04	0.00E+00	10.86
100	CROP100	0	9684	0.209021E+02	0.12E+02	0.00E+00	19.42

Tab. A.6: Detailed Results of MISQPOA for the Academic Test Set

The results of the new outer approximation method MIQPSOA are reviewed in detail in Table A.7.

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	MITP1	0	289	-0.100097E+05	0.00E+00	0.00E+00	0.00
2	QIP1	0	39	-0.200000E+02	0.00E+00	0.00E+00	0.00
3	MITP2	0	48	0.350000E+01	-0.47E-10	0.82E-10	0.00
4	ASAADI11	0	217	-0.409578E+02	-0.28E-04	0.63E-05	0.00
5	ASAADI12	0	104	-0.380000E+02	0.00E+00	0.00E+00	0.00
6	ASAADI21	0	380	0.694903E+03	0.37E-08	0.00E+00	0.00
7	ASAADI22	0	257	0.700000E+03	0.00E+00	0.00E+00	0.01
8	ASAADI31	0	3923	0.372190E+02	-0.14E-04	0.00E+00	0.13
9	ASAADI32	0	877	0.430000E+02	0.00E+00	0.00E+00	0.40
10	DIRTY	0	44671	-0.304575E+01	0.49E-03	0.00E+00	2.92
11	BRAAK1	0	7251	0.100000E+01	0.91E-06	0.13E-14	0.08
12	BRAAK2	0	1581	-0.271811E+01	0.64E-04	0.00E+00	0.03
13	BRAAK3	0	746	-0.107199E+00	0.45E+00	0.00E+00	0.01
14	DEX2	0	66	-0.569375E+02	0.00E+00	0.00E+00	0.00
15	TP83	0	117	-0.306655E+05	-0.12E-10	0.86E-10	0.00
16	WP02	0	47	-0.244444E+01	-0.18E-05	0.00E+00	0.00
17	NVS01	0	112	0.167339E+02	0.34E+00	0.45E-12	0.00
18	NVS02	0	485	0.605795E+01	0.16E-01	0.74E-12	0.01
19	NVS03	0	58	0.160000E+02	0.00E+00	0.00E+00	0.00
20	NVS04	0	104	0.720000E+00	0.83E-14	0.00E+00	0.00
21	NVS05	0	790	0.775390E+01	0.42E+00	0.84E-04	0.03
22	NVS06	0	67	0.177031E+01	0.28E-06	0.00E+00	0.00
23	NVS07	0	48	0.400000E+01	0.00E+00	0.00E+00	0.00
24	NVS08	0	136	0.240126E+02	0.24E-01	0.00E+00	0.00
25	NVS09	0	94	-0.431343E+02	0.71E-07	0.00E+00	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
26	NVS10	0	110	-0.310800E+03	-0.18E-15	0.00E+00	0.00
27	NVS11	0	238	-0.431000E+03	0.00E+00	0.00E+00	0.01
28	NVS12	0	452	-0.481200E+03	0.00E+00	0.00E+00	0.01
29	NVS13	0	740	-0.585200E+03	0.00E+00	0.00E+00	0.02
30	NVS14	0	615	-0.403582E+05	-0.12E-06	0.19E-13	0.01
31	NVS15	0	65	0.100000E+01	0.00E+00	0.00E+00	0.00
32	NVS16	0	64	0.703125E+00	0.00E+00	0.00E+00	0.00
33	NVS17	0	3134	-0.110040E+04	-0.21E-15	0.00E+00	0.38
34	NVS18	0	832	-0.778400E+03	0.00E+00	0.00E+00	0.08
35	NVS19	0	2701	-0.109840E+04	0.00E+00	0.00E+00	0.61
36	NVS20	0	1306	0.230922E+03	-0.21E-07	0.22E-15	0.04
37	NVS21	0	120	-0.128640E-02	0.10E+01	0.00E+00	0.00
38	NVS22	4	213	0.644726E+02	0.96E+01	0.70E-02	0.02
39	NVS23	0	4300	-0.112520E+04	0.00E+00	0.00E+00	2.41
40	NVS24	0	3826	-0.102380E+04	0.91E-02	0.00E+00	1.12
41	GEAR	0	138	0.100001E+01	0.14E-04	0.00E+00	0.00
42	GEAR2	0	928	0.100636E+01	0.64E-02	0.00E+00	0.05
43	GEAR3	0	255	0.100001E+01	0.57E-05	0.39E-11	0.00
44	GEAR4	0	113	0.000000E+00	-0.10E+01	0.57E-04	0.00
45	WINDFAC	0	1116	0.750000E+00	0.19E+01	0.90E-09	0.03
46	DG1	0	796	0.600871E+01	-0.17E-03	0.13E-04	0.02
47	DG2	0	124	0.172142E+02	0.27E-05	0.22E-10	0.00
48	DG3	0	3135	0.680097E+02	-0.38E-05	0.00E+00	0.33
49	FLOUDAS1	0	187	0.793111E+01	0.34E-01	0.00E+00	0.00
50	FLOUDAS2	0	27	0.107654E+01	0.29E-05	0.00E+00	0.00
51	FLOUDAS3	0	181	0.457958E+01	-0.26E-07	0.11E-07	0.00
52	FLOUDAS4	0	570	-0.824670E+00	0.13E+00	0.12E-06	0.04
53	FLOUDAS5	0	24	0.310000E+02	0.00E+00	0.00E+00	0.00
54	FLOUDAS6	0	24	-0.170000E+02	-0.39E-12	0.13E-11	0.00
55	OAER	0	241	-0.192347E+01	-0.19E-03	0.32E-04	0.00
56	SPRING	0	1283	0.126241E+01	0.49E+00	0.70E-04	0.11
57	DAKOTA	0	218	0.136221E+01	-0.87E-03	0.86E-04	0.00
58	PROB02	0	198	0.112235E+06	0.00E+00	0.00E+00	0.00
59	PROB03	0	25	0.100000E+02	0.00E+00	0.00E+00	0.00
60	PROB10	0	26	0.344550E+01	-0.46E-11	0.41E-11	0.00
61	BATCH	0	5889	0.333240E+00	0.17E+00	0.14E-04	6.35
62	BATCHDES	0	159	0.239954E+06	0.43E+00	0.74E-04	0.00
63	DU_OPT5	0	3949	0.258716E+02	0.22E+01	0.00E+00	0.22
64	DU_OPT	0	7498	0.606938E+01	0.71E+00	0.00E+00	0.49
65	ST_E13	0	17	0.223607E+01	0.12E+00	0.00E+00	0.00
66	ST_E32	4	1131	-0.405178E+02	-0.27E+02	0.50E+00	0.34
67	ST_E36	0	310	-0.244843E+03	0.47E-02	0.14E-05	0.01
68	ST_E38	0	174	0.719773E+04	0.60E-12	0.38E-14	0.00
69	ST_MIQP1	0	39	0.281000E+03	0.00E+00	0.00E+00	0.00
70	ST_MIQP2	0	81	0.200000E+01	0.00E+00	0.00E+00	0.00
71	ST_MIQP3	0	51	-0.600000E+01	0.00E+00	0.00E+00	0.00
72	ST_MIQP4	0	63	-0.457400E+04	-0.82E-12	0.24E-10	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
73	ST_MIQP5	0	124	-0.333889E+03	0.33E-07	0.88E-12	0.00
74	ST_TEST1	0	22	0.100000E+01	0.00E+00	0.00E+00	0.00
75	ST_TEST2	0	33	-0.925000E+01	0.00E+00	0.00E+00	0.00
76	ST_TEST3	0	113	-0.700000E+01	0.00E+00	0.00E+00	0.00
77	ST_TEST4	0	80	-0.700000E+01	0.00E+00	0.00E+00	0.00
78	ST_TEST5	0	120	-0.110000E+03	0.00E+00	0.00E+00	0.00
79	ST_TEST6	0	85	0.471000E+03	0.00E+00	0.00E+00	0.00
80	ST_TEST8	0	752	-0.296050E+05	0.00E+00	0.00E+00	0.10
81	ST_TESTGR1	0	974	-0.128116E+02	-0.14E-15	0.00E+00	0.25
82	ST_TESTGR3	0	1664	-0.205900E+02	0.00E+00	0.00E+00	1.19
83	ST_TESTPH4	0	26	-0.805000E+02	0.00E+00	0.00E+00	0.00
84	TLN2	0	111	0.530000E+01	0.00E+00	0.00E+00	0.02
85	TLN4	0	268	0.100000E+02	0.20E+00	0.00E+00	3.66
86	TLN5	0	326	0.140000E+02	0.36E+00	0.00E+00	7.19
87	TLN6	0	520	0.171000E+02	0.12E+00	0.00E+00	12.25
88	PROCSEL	0	379	-0.141101E+01	0.27E+00	0.68E-05	0.02
89	TLOSS	0	681	0.181000E+02	0.11E+00	0.00E+00	22.97
90	TLTR	0	289	0.480667E+02	-0.74E-15	0.00E+00	6.47
91	ALAN	0	167	0.292500E+01	0.90E-02	0.30E-10	0.00
92	MEANVARX	0	1269	0.141897E+02	-0.12E-01	0.10E-13	0.10
93	HMITTELMANN	0	101	0.500000E+02	0.28E+01	0.00E+00	0.02
94	MIP_EX	0	25	0.350000E+01	0.00E+00	0.00E+00	0.00
95	MGRID_CYCLES1	0	54	0.800000E+01	0.00E+00	0.00E+00	0.00
96	MGRID_CYCLES2	0	883	0.310000E+03	0.33E-01	0.44E-05	0.05
97	CROP5	0	522	0.100409E+00	-0.51E-05	0.00E+00	0.03
98	CROP20	0	637	0.661098E+00	0.40E+01	0.00E+00	5.55
99	CROP50	0	1992	0.174059E+01	0.36E+01	0.00E+00	61.55
100	CROP100	0	2063	-0.102991E+03	-0.67E+02	0.00E+00	183.45

Tab. A.7: Detailed Results of MIQPSOA for the Academic Test Set

The detailed results of the code MIQPSOA without performing mixed-integer search steps, i.e., applying well-known linear outer approximatin techniques, are shown in Table A.8.

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	MITP1	0	550	-0.786540E+04	0.21E+00	0.00E+00	0.00
2	QIP1	0	55	-0.100000E+02	0.50E+00	0.00E+00	0.00
3	MITP2	0	48	0.350000E+01	-0.47E-10	0.82E-10	0.00
4	ASAADI11	0	329	-0.409574E+02	-0.20E-04	0.30E-09	0.00
5	ASAADI12	0	116	-0.380000E+02	0.00E+00	0.00E+00	0.00
6	ASAADI21	0	531	0.694903E+03	0.37E-08	0.00E+00	0.00
7	ASAADI22	0	496	0.700000E+03	0.00E+00	0.00E+00	0.01
8	ASAADI31	0	3873	0.372190E+02	-0.14E-04	0.00E+00	0.07
9	ASAADI32	0	911	0.430000E+02	0.00E+00	0.00E+00	0.36
10	DIRTY	0	43090	-0.304574E+01	0.49E-03	0.00E+00	0.79
11	BRAAK1	0	9909	0.100000E+01	0.66E-06	0.19E-13	0.05
12	BRAAK2	0	1929	-0.271828E+01	-0.27E-06	0.12E-09	0.01
13	BRAAK3	0	748	-0.104979E+00	0.47E+00	0.00E+00	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
14	DEX2	0	85	-0.569375E+02	0.00E+00	0.00E+00	0.00
15	TP83	0	247	-0.306655E+05	-0.12E-10	0.86E-10	0.00
16	WP02	0	88	-0.244444E+01	-0.18E-05	0.00E+00	0.00
17	NVS01	0	66	0.491987E+03	0.38E+02	0.14E-09	0.00
18	NVS02	4	131	0.596418E+01	-0.80E-07	0.10E+01	0.00
19	NVS03	0	76	0.160000E+02	0.00E+00	0.00E+00	0.00
20	NVS04	0	100	0.720000E+00	0.83E-14	0.00E+00	0.00
21	NVS05	0	1470	0.588728E+01	0.76E-01	0.97E-04	0.01
22	NVS06	0	76	0.177031E+01	0.28E-06	0.00E+00	0.00
23	NVS07	0	34	0.400000E+01	0.00E+00	0.00E+00	0.00
24	NVS08	0	410	0.234497E+02	-0.87E-06	0.48E-05	0.00
25	NVS09	0	94	-0.431343E+02	0.71E-07	0.00E+00	0.00
26	NVS10	0	111	-0.310800E+03	-0.18E-15	0.00E+00	0.00
27	NVS11	0	269	-0.431000E+03	0.00E+00	0.00E+00	0.00
28	NVS12	0	511	-0.481200E+03	0.00E+00	0.00E+00	0.01
29	NVS13	0	431	-0.542200E+03	0.73E-01	0.00E+00	0.01
30	NVS14	0	413	-0.374822E+05	0.71E-01	0.16E-12	0.00
31	NVS15	0	86	0.100000E+01	0.00E+00	0.00E+00	0.00
32	NVS16	0	62	0.703125E+00	0.00E+00	0.00E+00	0.00
33	NVS17	0	234	-0.392800E+03	0.64E+00	0.00E+00	0.01
34	NVS18	0	886	-0.778400E+03	0.00E+00	0.00E+00	0.08
35	NVS19	0	999	-0.988200E+03	0.10E+00	0.00E+00	0.50
36	NVS20	0	1499	0.230922E+03	-0.21E-07	0.00E+00	0.02
37	NVS21	0	215	-0.490860E+01	0.14E+00	0.00E+00	0.00
38	NVS22	0	239	0.822135E+01	0.36E+00	0.14E-05	0.00
39	NVS23	0	154	-0.111400E+03	0.90E+00	0.00E+00	0.00
40	NVS24	0	259	-0.118000E+03	0.89E+00	0.00E+00	0.03
41	GEAR	0	138	0.100001E+01	0.14E-04	0.00E+00	0.00
42	GEAR2	0	1052	0.100636E+01	0.64E-02	0.49E-11	0.01
43	GEAR3	0	255	0.100001E+01	0.57E-05	0.39E-11	0.00
44	GEAR4	0	324	0.000000E+00	-0.10E+01	0.57E-04	0.01
45	WINDFAC	4	312	0.443763E+02	0.17E+03	0.27E+00	0.00
46	DG1	0	172	0.600876E+01	-0.16E-03	0.32E-07	0.00
47	DG2	0	361	0.426835E+02	0.15E+01	0.21E-10	0.00
48	DG3	0	1471	0.680097E+02	-0.38E-05	0.70E-07	0.03
49	FLOUDAS1	0	186	0.793111E+01	0.34E-01	0.00E+00	0.00
50	FLOUDAS2	0	26	0.107654E+01	0.29E-05	0.00E+00	0.00
51	FLOUDAS3	0	197	0.457958E+01	-0.26E-07	0.11E-07	0.00
52	FLOUDAS4	0	525	-0.943472E+00	-0.21E-05	0.54E-04	0.00
53	FLOUDAS5	0	26	0.310000E+02	0.00E+00	0.00E+00	0.00
54	FLOUDAS6	0	33	-0.170000E+02	-0.51E-11	0.17E-10	0.00
55	OAER	0	727	-0.192310E+01	-0.22E-05	0.35E-06	0.00
56	SPRING	4	247	0.125393E+00	-0.85E+00	0.27E+00	0.00
57	DAKOTA	0	146	0.136340E+01	0.00E+00	0.00E+00	0.00
58	PROB02	0	71	0.368285E+06	0.23E+01	0.00E+00	0.00
59	PROB03	0	25	0.100000E+02	0.00E+00	0.00E+00	0.00
60	PROB10	0	61	0.344550E+01	-0.12E-11	0.21E-12	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
61	BATCH	0	17769	0.344933E+00	0.21E+00	0.20E-04	119.46
62	BATCHDES	0	297	0.239958E+06	0.43E+00	0.31E-04	0.00
63	DU_OPT5	0	4483	0.258716E+02	0.22E+01	0.00E+00	0.06
64	DU_OPT	0	7785	0.760374E+01	0.11E+01	0.00E+00	0.10
65	ST_E13	0	17	0.223607E+01	0.12E+00	0.00E+00	0.00
66	ST_E32	4	6291	-0.222519E+01	-0.56E+00	0.85E+01	0.02
67	ST_E36	0	127	-0.197217E+03	0.20E+00	0.90E-04	0.00
68	ST_E38	0	321	0.719773E+04	0.60E-12	0.38E-14	0.00
69	ST_MIQP1	0	39	0.281000E+03	0.00E+00	0.00E+00	0.00
70	ST_MIQP2	0	71	0.200000E+01	0.00E+00	0.00E+00	0.00
71	ST_MIQP3	0	32	-0.600000E+01	0.00E+00	0.00E+00	0.00
72	ST_MIQP4	0	44	-0.457400E+04	0.00E+00	0.00E+00	0.00
73	ST_MIQP5	0	205	-0.333889E+03	0.33E-07	0.22E-15	0.00
74	ST_TEST1	0	22	0.100000E+01	0.00E+00	0.00E+00	0.00
75	ST_TEST2	0	46	-0.925000E+01	0.00E+00	0.00E+00	0.00
76	ST_TEST3	0	97	-0.700000E+01	0.00E+00	0.00E+00	0.00
77	ST_TEST4	0	79	-0.700000E+01	0.00E+00	0.00E+00	0.00
78	ST_TEST5	0	75	-0.110000E+03	0.00E+00	0.00E+00	0.00
79	ST_TEST6	0	74	0.471000E+03	0.00E+00	0.00E+00	0.00
80	ST_TEST8	4	147	0.276831E+07	0.95E+02	0.49E+02	0.00
81	ST_TESTGR1	0	974	-0.128116E+02	-0.14E-15	0.00E+00	0.23
82	ST_TESTGR3	0	701	-0.205900E+02	0.00E+00	0.00E+00	0.69
83	ST_TESTPH4	0	35	-0.805000E+02	0.00E+00	0.00E+00	0.00
84	TLN2	0	81	0.530000E+01	0.00E+00	0.00E+00	0.00
85	TLN4	4	139	0.400000E+01	-0.52E+00	0.10E+01	1.52
86	TLN5	4	203	0.500000E+01	-0.51E+00	0.10E+01	1.85
87	TLN6	4	279	0.600000E+01	-0.61E+00	0.10E+01	2.74
88	PROCSEL	0	1272	-0.192310E+01	0.50E-07	0.78E-07	0.01
89	TLOSS	4	279	0.600000E+01	-0.63E+00	0.25E+04	4.34
90	TLTR	4	267	0.350000E+02	-0.27E+00	0.12E+03	5.21
91	ALAN	4	141	0.703141E+01	0.14E+01	0.27E-01	0.00
92	MEANVARX	0	724	0.235829E+02	0.64E+00	0.56E-16	0.08
93	HMITTELMANN	0	116	0.210000E+02	0.62E+00	0.00E+00	0.01
94	MIP_EX	0	51	0.350000E+01	0.00E+00	0.00E+00	0.00
95	MGRID_CYCLES1	0	71	0.800000E+01	0.00E+00	0.00E+00	0.00
96	MGRID_CYCLES2	0	473	0.310000E+03	0.33E-01	0.44E-05	0.03
97	CROP5	0	873	0.100409E+00	-0.51E-05	0.00E+00	0.06
98	CROP20	0	1569	0.404518E+00	0.21E+01	0.00E+00	10.32
99	CROP50	0	2370	0.169215E+01	0.35E+01	0.00E+00	99.29
100	CROP100	0	2063	-0.102461E+03	-0.66E+02	0.00E+00	181.62

Tab. A.8: Detailed Results of a Linear Outer Approximation Method for the Academic Test Set

The results of the NLP-based branch-and-bound method MINLPB4 are presented in Table A.9.

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	MITP1	0	144	-0.100097E+05	0.42E-14	0.00E+00	0.00
2	QIP1	0	35	-0.200000E+02	0.00E+00	0.00E+00	0.00
3	MITP2	0	163	0.350000E+01	-0.83E-07	0.83E-07	0.00
4	ASAADI11	0	96	-0.409574E+02	-0.20E-04	0.18E-10	0.00
5	ASAADI12	0	344	-0.380000E+02	0.00E+00	0.00E+00	0.00
6	ASAADI21	0	832	0.694903E+03	0.28E-07	0.00E+00	0.00
7	ASAADI22	0	2118	0.700000E+03	0.00E+00	0.00E+00	0.01
8	ASAADI31	0	4893	0.372190E+02	-0.14E-04	0.40E-13	0.01
9	ASAADI32	0	16938	0.430000E+02	0.00E+00	0.00E+00	0.04
10	DIRTY	0	9880	-0.304680E+01	0.15E-03	0.00E+00	0.06
11	BRAAK1	0	763	0.344800E+01	0.24E+01	0.00E+00	0.00
12	BRAAK2	0	2698	-0.935872E+00	0.66E+00	0.00E+00	0.00
13	BRAAK3	0	538	-0.196176E+00	0.20E-02	0.00E+00	0.00
14	DEX2	0	60	-0.569375E+02	0.00E+00	0.00E+00	0.00
15	TP83	0	186	-0.306655E+05	0.47E-06	0.18E-06	0.00
16	WP02	0	34	-0.244444E+01	-0.18E-05	0.00E+00	0.00
17	NVS01	0	240	0.124697E+02	-0.95E-07	0.58E-14	0.00
18	NVS02	0	1045	0.596418E+01	-0.80E-07	0.13E-12	0.00
19	NVS03	0	54	0.160000E+02	0.00E+00	0.00E+00	0.00
20	NVS04	0	345	0.720000E+00	0.83E-14	0.00E+00	0.00
21	NVS05	0	548	0.547093E+01	0.20E-07	0.20E-09	0.00
22	NVS06	0	114	0.177031E+01	0.28E-06	0.00E+00	0.00
23	NVS07	0	45	0.400000E+01	0.00E+00	0.00E+00	0.00
24	NVS08	0	149	0.234497E+02	-0.11E-06	0.29E-10	0.00
25	NVS09	0	55	-0.431343E+02	0.71E-07	0.00E+00	0.00
26	NVS10	0	53	-0.310800E+03	-0.18E-15	0.00E+00	0.00
27	NVS11	0	147	-0.431000E+03	0.00E+00	0.00E+00	0.00
28	NVS12	0	263	-0.481200E+03	0.00E+00	0.00E+00	0.00
29	NVS13	0	740	-0.585200E+03	0.00E+00	0.00E+00	0.00
30	NVS14	0	1747	-0.403582E+05	-0.12E-06	0.20E-12	0.00
31	NVS15	0	112	0.100000E+01	0.00E+00	0.00E+00	0.00
32	NVS16	0	57	0.703125E+00	0.00E+00	0.00E+00	0.00
33	NVS17	0	4058	-0.110040E+04	-0.21E-15	0.00E+00	0.01
34	NVS18	0	2751	-0.778400E+03	0.00E+00	0.00E+00	0.00
35	NVS19	0	6292	-0.109840E+04	0.00E+00	0.00E+00	0.02
36	NVS20	0	2635	0.230922E+03	-0.20E-07	0.60E-13	0.01
37	NVS21	0	4	-0.201000E-04	0.10E+01	0.00E+00	0.00
38	NVS22	0	598	0.605822E+01	0.00E+00	0.28E-06	0.00
39	NVS23	0	8897	-0.112520E+04	0.00E+00	0.00E+00	0.03
40	NVS24	0	13465	-0.102380E+04	0.91E-02	0.00E+00	0.05
41	GEAR	0	90	0.100000E+01	0.78E-06	0.00E+00	0.00
42	GEAR2	0	3484	0.100003E+01	0.31E-04	0.00E+00	0.01
43	GEAR3	0	333	0.100000E+01	0.78E-06	0.72E-11	0.00
44	GEAR4	3	345189	0.000000E+00	-0.10E+01	0.10E+01	0.66
45	WINDFAC	0	843	0.254487E+00	-0.65E-08	0.10E-12	0.01
46	DG1	0	212	0.600876E+01	-0.16E-03	0.79E-09	0.00
47	DG2	0	636	0.172142E+02	0.27E-05	0.00E+00	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
48	DG3	0	1930	0.680097E+02	-0.37E-05	0.10E-12	0.02
49	FLOUDAS1	0	55	0.766718E+01	-0.39E-08	0.11E-08	0.00
50	FLOUDAS2	0	68	0.107654E+01	0.29E-05	0.00E+00	0.00
51	FLOUDAS3	0	408	0.457958E+01	-0.60E-07	0.85E-07	0.00
52	FLOUDAS4	0	1322	-0.943471E+00	-0.34E-08	0.12E-06	0.00
53	FLOUDAS5	0	42	0.310000E+02	0.00E+00	0.00E+00	0.00
54	FLOUDAS6	0	12	-0.170000E+02	0.18E-10	0.00E+00	0.00
55	OAER	0	290	-0.192310E+01	0.24E-06	0.22E-10	0.00
56	SPRING	0	2431	0.846246E+00	-0.37E-07	0.20E-08	0.01
57	DAKOTA	0	85	0.136340E+01	0.00E+00	0.00E+00	0.00
58	PROB02	0	141	0.112235E+06	0.78E-13	0.00E+00	0.00
59	PROB03	0	48	0.100000E+02	0.00E+00	0.00E+00	0.00
60	PROB10	0	43	0.344550E+01	0.18E-10	0.00E+00	0.00
61	BATCH	0	17190	0.285506E+00	0.17E-05	0.10E-07	0.37
62	BATCHDES	0	100	0.239960E+06	0.43E+00	0.54E-07	0.00
63	DU_OPT5	0	37456	0.174526E+02	0.12E+01	0.00E+00	0.33
64	DU_OPT	0	22312	0.751195E+01	0.11E+01	0.00E+00	0.20
65	ST_E13	0	15	0.223607E+01	0.12E+00	0.00E+00	0.00
66	ST_E32	0	51793	-0.143041E+01	0.12E-07	0.16E-09	0.60
67	ST_E36	0	168	-0.166444E+03	0.32E+00	0.39E-14	0.00
68	ST_E38	0	111	0.719773E+04	0.60E-12	0.00E+00	0.00
69	ST_MIQP1	0	574	0.281000E+03	0.00E+00	0.00E+00	0.00
70	ST_MIQP2	0	146	0.200000E+01	-0.21E-06	0.17E-07	0.00
71	ST_MIQP3	0	27	-0.600000E+01	-0.65E-14	0.13E-13	0.00
72	ST_MIQP4	0	175	-0.457400E+04	-0.24E-10	0.57E-09	0.00
73	ST_MIQP5	0	96	-0.333889E+03	0.33E-07	0.97E-11	0.00
74	ST_TEST1	0	165	0.100000E+01	0.00E+00	0.00E+00	0.00
75	ST_TEST2	0	63	-0.925000E+01	0.00E+00	0.00E+00	0.00
76	ST_TEST3	0	673	-0.700000E+01	-0.13E-11	0.89E-11	0.00
77	ST_TEST4	0	98	-0.700000E+01	0.47E-10	0.00E+00	0.00
78	ST_TEST5	0	3061	-0.110000E+03	0.00E+00	0.00E+00	0.02
79	ST_TEST6	0	4017	0.471000E+03	0.00E+00	0.00E+00	0.02
80	ST_TEST8	0	436	-0.296050E+05	-0.37E-11	0.21E-10	0.01
81	ST_TESTGR1	0	9061	-0.128116E+02	-0.28E-13	0.99E-13	0.02
82	ST_TESTGR3	0	73716	-0.205900E+02	0.00E+00	0.00E+00	0.31
83	ST_TESTPH4	0	48	-0.805000E+02	0.00E+00	0.00E+00	0.00
84	TLN2	0	3779	0.530000E+01	-0.22E-11	0.58E-11	0.01
85	TLN4	1	2720887	0.860000E+01	0.36E-01	0.49E-31	13.79
86	TLN5	1	4229766	0.106000E+02	0.29E-01	0.00E+00	26.12
87	TLN6	1	5707675	0.159000E+02	0.39E-01	0.56E-10	50.69
88	PROCSEL	0	630	-0.192310E+01	0.14E-06	0.71E-11	0.00
89	TLOSS	0	38934	0.163000E+02	-0.76E-10	0.79E-08	0.48
90	TLTR	2	2405	0.000000E+00	-0.10E+01	0.10E+01	0.04
91	ALAN	0	379	0.292500E+01	0.90E-02	0.18E-10	0.00
92	MEANVARX	0	1377	0.141897E+02	-0.12E-01	0.21E-11	0.01
93	HMITTELMANN	0	2097	0.130000E+02	0.00E+00	0.00E+00	0.01
94	MIP_EX	0	72	0.350000E+01	-0.35E-07	0.83E-07	0.00

TP	NAME	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
95	MGRID_CYCLES1	0	610	0.800000E+01	0.00E+00	0.00E+00	0.00
96	MGRID_CYCLES2	0	59222	0.300000E+03	0.00E+00	0.00E+00	0.18
97	CROP5	0	1345	0.100409E+00	-0.51E-05	0.00E+00	0.00
98	CROP20	0	452223	0.131785E+00	-0.17E-07	0.00E+00	4.23
99	CROP50	1	5312432	0.585701E+00	0.56E+00	0.00E+00	130.33
100	CROP100	0	101	-0.886964E+02	-0.58E+02	0.00E+00	0.01

Tab. A.9: Detailed Results of MINLPB4 for the Academic Test Set

A.2 Test Cases from Petroleum Industry

We considered a test set containing 55 optimization problems, that were provided by our cooperation partner shell. The test cases represent some of the problem characteristics arising in well relinking and lift gas applications, see Chapter 1 for further details. The subsequent table presents the problem characteristics and the best-known objective value. See also Table A.2 for further explanation.

TP	n_c	n_i	n_b	m_e	m	f^*
1	3	0	9	3	9	-0.16045000E+04
2	6	0	18	6	15	-0.18000000E+04
3	6	0	18	6	15	-0.23083000E+04
4	6	0	18	6	15	-0.25082919E+04
5	6	0	18	6	15	-0.26266000E+04
6	6	0	18	6	15	-0.28045000E+04
7	6	0	18	6	15	-0.30995000E+04
8	9	0	27	9	21	-0.26310000E+04
9	9	0	27	9	21	-0.30406261E+04
10	9	0	27	9	21	-0.34045000E+04
11	10	0	10	0	1	-0.15020820E+04
12	10	0	10	0	1	-0.30418740E+04
13	10	0	10	0	1	-0.45049970E+04
14	10	0	10	0	1	-0.59232000E+04
15	10	0	10	0	1	-0.72820330E+04
16	10	0	10	0	1	-0.85874180E+04
17	10	0	10	0	1	-0.97723790E+04
18	10	0	10	0	1	-0.10789980E+05
19	10	0	10	0	1	-0.11644540E+05
20	10	0	10	0	1	-0.12317110E+05
21	10	0	10	0	11	-0.22360680E+03
22	10	0	10	0	11	-0.31622780E+03
23	10	0	10	0	11	-0.38890870E+03
24	10	0	10	0	11	-0.46475800E+03
25	10	0	10	0	11	-0.54382900E+03
26	10	0	10	0	11	-0.62749500E+03
27	10	0	10	0	11	-0.71554180E+03

TP	n_c	n_i	n_b	m_e	m	f^*
28	10	0	10	0	11	-0.80638080E+03
29	10	0	10	0	11	-0.90124910E+03
30	10	0	10	0	11	-0.99636840E+03
31	10	0	10	0	11	-0.10831670E+04
32	10	0	10	0	11	-0.11635080E+04
33	10	0	10	0	11	-0.12386480E+04
34	10	0	10	0	11	-0.13094850E+04
35	10	0	10	0	11	-0.13766810E+04
36	10	0	10	0	11	-0.14407460E+04
37	10	0	10	0	11	-0.15021000E+04
38	10	0	10	0	11	-0.23041000E+04
39	10	0	10	0	11	-0.30419000E+04
40	10	0	10	0	11	-0.37610000E+04
41	10	0	10	0	11	-0.45050000E+04
42	10	0	10	0	11	-0.52231000E+04
43	10	0	10	0	11	-0.59232000E+04
44	10	0	10	0	11	-0.66091000E+04
45	10	0	10	0	11	-0.72820000E+04
46	10	0	10	0	11	-0.79417000E+04
47	10	0	10	0	11	-0.85874000E+04
48	10	0	10	0	11	-0.92014000E+04
49	10	0	10	0	11	-0.97724000E+04
50	10	0	10	0	11	-0.10301000E+05
51	10	0	10	0	11	-0.10790000E+05
52	10	0	10	0	11	-0.11238000E+05
53	10	0	10	0	11	-0.11645000E+05
54	10	0	10	0	11	-0.12008000E+05
55	10	0	10	0	11	-0.12317000E+05

Tab. A.10: Characteristics of Shell Test Cases

The subsequent Table A.11 presents detailed results of the solver MISQP for the petroleum test cases.

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	0	368	-0.16044944E+04	0.35E-05	0.86E-11	0.04
2	0	1125	-0.18000000E+04	-0.11E-11	0.81E-11	0.80
3	0	1411	-0.23082919E+04	0.35E-05	0.85E-10	1.03
4	0	3299	-0.25082919E+04	0.20E-07	0.54E-11	2.06
5	0	1811	-0.26249297E+04	0.64E-03	0.65E-11	1.24
6	0	1506	-0.28044944E+04	0.20E-05	0.46E-11	0.84
7	0	733	-0.30995309E+04	-0.10E-04	0.22E-09	0.32
8	0	6396	-0.25090878E+04	0.46E-01	0.37E-10	22.45
9	0	4493	-0.30254621E+04	0.50E-02	0.33E-10	16.75
10	1	74084	-0.33644979E+04	0.12E-01	0.32E-10	22.86
11	0	597	-0.15020819E+04	0.74E-07	0.00E+00	0.04
12	0	532	-0.30418744E+04	-0.14E-06	0.58E-08	0.04
13	0	2115	-0.45049972E+04	-0.50E-07	0.00E+00	0.23

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
14	0	1225	-0.59231537E+04	0.78E-05	0.00E+00	0.09
15	0	1940	-0.72663608E+04	0.22E-02	0.00E+00	0.17
16	0	453	-0.85874181E+04	-0.14E-07	0.00E+00	0.02
17	0	487	-0.97723791E+04	-0.13E-07	0.00E+00	0.01
18	0	549	-0.10789985E+05	-0.44E-06	0.00E+00	0.01
19	0	484	-0.11644542E+05	-0.21E-06	0.00E+00	0.01
20	0	42	-0.12317111E+05	-0.96E-07	0.00E+00	0.00
21	0	3071	-0.22360680E+03	-0.19E-09	0.51E-08	0.70
22	0	7511	-0.31622800E+03	-0.63E-06	0.59E-06	2.03
23	0	4042	-0.38890877E+03	-0.19E-06	0.97E-07	1.36
24	0	3561	-0.46475800E+03	-0.47E-08	0.12E-08	1.38
25	0	4141	-0.54221767E+03	0.30E-02	0.75E-09	1.56
26	0	2021	-0.62749510E+03	-0.15E-06	0.94E-07	0.92
27	0	2787	-0.71554175E+03	0.66E-07	0.75E-09	1.12
28	0	1265	-0.80498447E+03	0.17E-02	0.75E-09	0.71
29	0	1985	-0.90000000E+03	0.14E-02	0.75E-09	1.18
30	0	1619	-0.99636842E+03	-0.18E-07	0.11E-07	0.83
31	0	2791	-0.10831667E+04	0.31E-06	0.49E-08	1.28
32	0	2443	-0.11635076E+04	0.32E-06	0.73E-09	1.08
33	0	2701	-0.12386485E+04	-0.38E-06	0.16E-07	1.17
34	0	2139	-0.13094846E+04	0.28E-06	0.73E-09	0.98
35	0	3227	-0.13766808E+04	0.15E-06	0.73E-09	1.56
36	0	6272	-0.13565595E+04	0.58E-01	0.89E-06	4.55
37	0	1500	-0.14560231E+04	0.31E-01	0.59E-06	0.68
38	0	1880	-0.23040726E+04	0.12E-04	0.54E-07	1.24
39	0	2252	-0.30418746E+04	0.84E-05	0.45E-07	2.09
40	0	2106	-0.37609839E+04	0.43E-05	0.64E-09	2.14
41	0	1676	-0.44728364E+04	0.71E-02	0.37E-06	0.64
42	0	8839	-0.52231478E+04	-0.92E-05	0.46E-06	2.14
43	0	2740	-0.59089773E+04	0.24E-02	0.19E-06	0.84
44	0	3358	-0.66090883E+04	0.18E-05	0.85E-06	0.60
45	0	2934	-0.72727074E+04	0.13E-02	0.24E-06	0.51
46	0	2706	-0.79417410E+04	-0.52E-05	0.43E-08	0.31
47	0	1980	-0.85874196E+04	-0.23E-05	0.22E-06	0.20
48	0	1743	-0.92014344E+04	-0.37E-05	0.27E-06	0.17
49	0	1743	-0.97723804E+04	0.20E-05	0.17E-06	0.15
50	0	1657	-0.10301601E+05	-0.58E-04	0.32E-06	0.13
51	0	1214	-0.10789986E+05	0.13E-05	0.29E-06	0.09
52	0	680	-0.11237843E+05	0.14E-04	0.60E-09	0.05
53	0	549	-0.11644542E+05	0.39E-04	0.73E-08	0.03
54	0	421	-0.12007632E+05	0.31E-04	0.20E-06	0.02
55	0	42	-0.12317111E+05	-0.90E-05	0.00E+00	0.00

Tab. A.11: Detailed Results of MISQP for the Shell Test Set

We review the results of the basic version of MISQP for the industrial test cases in Table A.12.

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	0	91	-0.16044944E+04	0.35E-05	0.86E-11	0.00
2	4	172	-0.14203822E+04	0.21E+00	0.37E+00	0.21
3	0	325	-0.23082919E+04	0.35E-05	0.41E-11	0.17
4	0	326	-0.23066960E+04	0.80E-01	0.59E-11	0.17
5	0	225	-0.26249297E+04	0.64E-03	0.65E-11	0.20
6	0	200	-0.28044944E+04	0.20E-05	0.16E-10	0.07
7	0	350	-0.30995309E+04	-0.10E-04	0.22E-09	0.13
8	0	1148	-0.24073929E+04	0.85E-01	0.46E-11	1.82
9	0	631	-0.27037975E+04	0.11E+00	0.31E-10	5.93
10	0	741	-0.31016949E+04	0.89E-01	0.81E-11	1.65
11	0	146	-0.15020819E+04	0.74E-07	0.00E+00	0.01
12	0	275	-0.30418744E+04	-0.14E-06	0.58E-08	0.02
13	0	253	-0.43499808E+04	0.34E-01	0.00E+00	0.03
14	0	896	-0.59089762E+04	0.24E-02	0.00E+00	0.07
15	0	1152	-0.70493335E+04	0.32E-01	0.00E+00	0.11
16	0	453	-0.85874181E+04	-0.14E-07	0.00E+00	0.02
17	0	487	-0.97723791E+04	-0.13E-07	0.00E+00	0.01
18	0	549	-0.10789985E+05	-0.44E-06	0.00E+00	0.01
19	0	484	-0.11644542E+05	-0.21E-06	0.00E+00	0.01
20	0	42	-0.12317111E+05	-0.96E-07	0.00E+00	0.00
21	0	849	-0.22360680E+03	-0.19E-09	0.51E-08	0.18
22	0	1191	-0.20554840E+03	0.35E+00	0.34E-06	0.32
23	0	1144	-0.35601993E+03	0.85E-01	0.38E-06	0.43
24	0	893	-0.46016301E+03	0.99E-02	0.75E-09	0.61
25	0	550	-0.46556417E+03	0.14E+00	0.75E-09	0.26
26	0	717	-0.61967734E+03	0.12E-01	0.75E-09	0.54
27	0	787	-0.71116102E+03	0.61E-02	0.75E-09	0.58
28	0	530	-0.80498447E+03	0.17E-02	0.75E-09	0.41
29	0	764	-0.89148754E+03	0.11E-01	0.17E-07	0.58
30	0	547	-0.77459667E+03	0.22E+00	0.74E-09	0.44
31	0	857	-0.93367637E+03	0.14E+00	0.70E-06	0.59
32	0	572	-0.91747004E+03	0.21E+00	0.35E-06	0.38
33	0	767	-0.10446291E+04	0.16E+00	0.75E-09	0.36
34	0	896	-0.10770330E+04	0.18E+00	0.12E-08	0.50
35	0	934	-0.12041600E+04	0.13E+00	0.27E-06	0.42
36	0	682	-0.12414723E+04	0.14E+00	0.55E-06	0.54
37	0	828	-0.14321328E+04	0.47E-01	0.77E-06	0.37
38	0	997	-0.23040726E+04	0.12E-04	0.54E-07	0.49
39	0	908	-0.25872285E+04	0.15E+00	0.47E-07	0.45
40	0	867	-0.36663333E+04	0.25E-01	0.69E-08	0.44
41	0	863	-0.44728364E+04	0.71E-02	0.37E-06	0.31
42	0	907	-0.50852783E+04	0.26E-01	0.93E-06	0.42
43	0	991	-0.59089773E+04	0.24E-02	0.19E-06	0.29
44	0	843	-0.66090864E+04	0.21E-05	0.31E-06	0.15
45	0	948	-0.72727074E+04	0.13E-02	0.24E-06	0.14
46	0	764	-0.79258282E+04	0.20E-02	0.35E-07	0.07
47	0	763	-0.85874196E+04	-0.23E-05	0.22E-06	0.05

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
48	0	757	-0.92014344E+04	-0.37E-05	0.27E-06	0.07
49	0	655	-0.97723804E+04	0.20E-05	0.17E-06	0.04
50	0	673	-0.10301601E+05	-0.58E-04	0.32E-06	0.04
51	0	467	-0.10789986E+05	0.13E-05	0.29E-06	0.02
52	0	445	-0.11237843E+05	0.14E-04	0.60E-09	0.02
53	0	360	-0.11644542E+05	0.39E-04	0.73E-08	0.02
54	0	295	-0.12007632E+05	0.31E-04	0.20E-06	0.01
55	0	42	-0.12317111E+05	-0.90E-05	0.00E+00	0.00

Tab. A.12: Detailed Results of MISQP without Fine-tuning for the Shell Test Set

The subsequent table shows the detailed results of the code MISQPOA for the Shell test case library.

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	0	490	-0.16044944E+04	0.35E-05	0.36E-11	0.05
2	0	786	-0.18000000E+04	-0.67E-11	0.22E-10	1.08
3	0	1418	-0.23082919E+04	0.35E-05	0.15E-10	1.90
4	0	2028	-0.25082920E+04	-0.26E-07	0.83E-07	2.33
5	0	1029	-0.26246799E+04	0.73E-03	0.83E-07	0.74
6	0	1022	-0.28044944E+04	0.20E-05	0.54E-11	0.99
7	0	574	-0.30995309E+04	-0.10E-04	0.81E-11	0.30
8	0	1611	-0.26310453E+04	-0.17E-04	0.32E-10	12.49
9	0	1690	-0.30315764E+04	0.30E-02	0.11E-10	6.30
10	0	2039	-0.33037975E+04	0.30E-01	0.24E-10	5.60
11	0	491	-0.15020819E+04	0.74E-07	0.00E+00	0.07
12	0	594	-0.29532883E+04	0.29E-01	0.00E+00	0.10
13	0	2699	-0.45049972E+04	-0.50E-07	0.00E+00	0.11
14	0	946	-0.59231537E+04	0.78E-05	0.00E+00	0.05
15	0	1429	-0.72820327E+04	0.44E-07	0.00E+00	0.12
16	0	406	-0.85874181E+04	-0.14E-07	0.00E+00	0.02
17	0	309	-0.97723791E+04	-0.13E-07	0.00E+00	0.01
18	0	1397	-0.10789985E+05	-0.44E-06	0.00E+00	0.03
19	0	165	-0.11644542E+05	-0.21E-06	0.00E+00	0.01
20	0	74	-0.12317111E+05	-0.96E-07	0.00E+00	0.00
21	0	3052	-0.22360680E+03	0.10E-07	0.21E-14	0.76
22	0	4722	-0.31622777E+03	0.11E-06	0.75E-09	1.12
23	0	5615	-0.38890873E+03	-0.76E-07	0.77E-14	1.33
24	0	13974	-0.46475800E+03	-0.33E-08	0.19E-14	2.58
25	0	11804	-0.54382902E+03	-0.32E-07	0.60E-13	2.87
26	0	16680	-0.62749502E+03	-0.32E-07	0.75E-09	5.59
27	0	15890	-0.71554175E+03	0.66E-07	0.56E-13	6.03
28	0	20262	-0.80638080E+03	-0.41E-08	0.20E-12	7.20
29	0	22318	-0.90124913E+03	-0.37E-07	0.17E-13	8.48
30	0	22202	-0.99636841E+03	-0.58E-08	0.23E-12	11.05
31	0	22137	-0.10831667E+04	0.32E-06	0.13E-12	12.92
32	0	27529	-0.11635076E+04	0.32E-06	0.45E-15	14.02

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
33	0	27650	-0.12386485E+04	-0.37E-06	0.73E-09	17.37
34	0	38466	-0.13094846E+04	0.28E-06	0.47E-15	31.26
35	0	37231	-0.13766808E+04	0.15E-06	0.73E-09	34.69
36	0	42456	-0.14407463E+04	-0.23E-06	0.32E-12	45.10
37	0	44927	-0.14797804E+04	0.15E-01	0.11E-13	42.55
38	0	47624	-0.23040725E+04	0.12E-04	0.33E-13	47.64
39	0	54610	-0.30418744E+04	0.84E-05	0.21E-12	55.60
40	0	59181	-0.37609839E+04	0.43E-05	0.26E-13	60.45
41	0	61213	-0.44728347E+04	0.71E-02	0.42E-13	66.54
42	0	64049	-0.51796477E+04	0.83E-02	0.34E-12	45.11
43	0	65739	-0.58882935E+04	0.59E-02	0.14E-11	35.61
44	0	63924	-0.65832171E+04	0.39E-02	0.57E-10	26.85
45	0	52269	-0.72663607E+04	0.21E-02	0.54E-10	14.04
46	0	37291	-0.79417409E+04	-0.52E-05	0.94E-13	4.40
47	0	30299	-0.85874180E+04	-0.21E-05	0.73E-13	3.01
48	0	20995	-0.92014325E+04	-0.35E-05	0.15E-13	1.26
49	0	9167	-0.97723791E+04	0.21E-05	0.66E-14	0.43
50	0	5028	-0.10301598E+05	-0.58E-04	0.69E-13	0.27
51	0	2836	-0.10789985E+05	0.14E-05	0.10E-12	0.12
52	0	1032	-0.11237843E+05	0.14E-04	0.16E-13	0.06
53	0	393	-0.11644542E+05	0.39E-04	0.36E-13	0.03
54	0	721	-0.12007631E+05	0.31E-04	0.13E-10	0.04
55	0	85	-0.12317111E+05	-0.90E-05	0.00E+00	0.00

Tab. A.13: Detailed Results of MISQPOA for the Shell Test Set

Table A.14 reviews the results of the new outer approximatin method MIQPSOA for the Shell test set in detail.

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	0	257	-0.16044944E+04	0.35E-05	0.00E+00	0.29
2	0	776	-0.91348656E+03	0.49E+00	0.98E-06	0.53
3	0	1567	-0.20044944E+04	0.13E+00	0.68E-11	0.42
4	0	500	-0.22044944E+04	0.12E+00	0.59E-11	0.30
5	0	502	-0.24044944E+04	0.85E-01	0.00E+00	0.31
6	0	1211	-0.23693519E+04	0.16E+00	0.00E+00	0.42
7	0	770	-0.30995309E+04	-0.10E-04	0.79E-15	0.31
8	0	1059	-0.25082919E+04	0.47E-01	0.11E-10	1.96
9	0	2006	-0.27876421E+04	0.83E-01	0.00E+00	1.53
10	0	1245	-0.31143884E+04	0.85E-01	0.38E-10	0.70
11	0	1126	-0.15020819E+04	0.74E-07	0.00E+00	0.25
12	0	1033	-0.30099834E+04	0.10E-01	0.17E-10	0.23
13	0	1440	-0.45049972E+04	-0.50E-07	0.00E+00	0.39
14	0	925	-0.59231537E+04	0.78E-05	0.00E+00	0.20
15	0	2284	-0.72820327E+04	0.43E-07	0.00E+00	0.29
16	0	1261	-0.85729954E+04	0.17E-02	0.00E+00	0.21
17	0	1386	-0.97723791E+04	-0.13E-07	0.00E+00	0.21
18	0	1491	-0.10789985E+05	-0.44E-06	0.36E-11	0.22

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
19	0	1155	-0.11644542E+05	-0.21E-06	0.00E+00	0.21
20	0	861	-0.12317111E+05	-0.96E-07	0.00E+00	0.20
21	0	1204	-0.22360680E+03	0.43E-08	0.29E-08	0.32
22	0	2847	-0.31622784E+03	-0.13E-06	0.19E-06	0.51
23	0	3305	-0.38890880E+03	-0.26E-06	0.16E-06	0.59
24	0	4134	-0.46475800E+03	-0.33E-08	0.15E-12	0.71
25	0	4841	-0.54382902E+03	-0.32E-07	0.28E-10	0.87
26	0	6874	-0.62749502E+03	-0.32E-07	0.10E-10	1.22
27	0	8570	-0.71554175E+03	0.66E-07	0.14E-11	1.83
28	0	12089	-0.80638088E+03	-0.97E-07	0.76E-07	3.14
29	0	14712	-0.90124926E+03	-0.18E-06	0.10E-06	5.78
30	0	19667	-0.99636888E+03	-0.48E-06	0.40E-06	10.91
31	0	9056	-0.10831672E+04	-0.22E-06	0.50E-06	1.94
32	0	21000	-0.11635076E+04	0.32E-06	0.32E-08	11.25
33	0	11190	-0.12386485E+04	-0.37E-06	0.20E-09	2.18
34	0	11248	-0.12181954E+04	0.70E-01	0.65E-08	2.24
35	0	33325	-0.13766808E+04	0.15E-06	0.93E-12	38.11
36	0	33355	-0.14407463E+04	-0.24E-06	0.46E-08	35.51
37	0	6491	-0.14474989E+04	0.36E-01	0.62E-06	0.90
38	0	10298	-0.22803510E+04	0.10E-01	0.84E-07	1.21
39	0	12083	-0.30418746E+04	0.83E-05	0.87E-07	2.09
40	0	21341	-0.37469990E+04	0.37E-02	0.43E-07	3.70
41	0	13377	-0.43416587E+04	0.36E-01	0.49E-08	1.69
42	0	22950	-0.51052915E+04	0.23E-01	0.27E-07	3.41
43	0	35907	-0.58326674E+04	0.15E-01	0.25E-06	8.01
44	0	11520	-0.64926747E+04	0.18E-01	0.34E-06	1.35
45	0	30998	-0.72300772E+04	0.71E-02	0.32E-06	3.96
46	1	40600	-0.79253199E+04	0.21E-02	0.15E-06	5.95
47	0	22858	-0.85874209E+04	-0.24E-05	0.85E-06	3.03
48	0	23974	-0.92014328E+04	-0.36E-05	0.79E-07	2.18
49	0	2100	-0.97723792E+04	0.21E-05	0.97E-08	0.41
50	0	13498	-0.10301599E+05	-0.58E-04	0.76E-07	1.10
51	0	5891	-0.10789986E+05	0.13E-05	0.48E-06	0.53
52	0	2585	-0.11237846E+05	0.14E-04	0.54E-06	0.33
53	0	1032	-0.11644544E+05	0.39E-04	0.23E-06	0.25
54	0	865	-0.12007635E+05	0.30E-04	0.93E-06	0.23
55	0	589	-0.12317111E+05	-0.90E-05	0.34E-16	0.20

Tab. A.14: Detailed Results of MIQPSOA for the Shell Test Set

Table A.15 reviews the results of the code MIQPSOA without performing mixed-integer search steps, i.e., applying a linear outer approximation method for solving the industrial test set in detail.

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	0	369	-0.64900000E+03	0.60E+00	0.00E+00	0.00
2	0	1322	-0.16591837E+04	0.78E-01	0.00E+00	0.02
3	4	561	-0.20000000E+01	0.10E+01	0.10E+01	0.01
4	0	2426	-0.23061893E+04	0.81E-01	0.79E-15	0.16

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
5	0	2401	-0.24044944E+04	0.85E-01	0.14E-09	0.18
6	0	2351	-0.28044944E+04	0.20E-05	0.79E-15	0.18
7	0	1272	-0.30995309E+04	-0.10E-04	0.44E-10	0.05
8	0	6550	-0.23061893E+04	0.12E+00	0.12E-10	2.23
9	0	6301	-0.30073929E+04	0.11E-01	0.00E+00	2.11
10	0	5793	-0.34044944E+04	0.17E-05	0.15E-15	1.32
11	0	703	-0.36767670E+03	0.76E+00	0.00E+00	0.00
12	0	657	-0.73618206E+03	0.76E+00	0.00E+00	0.00
13	0	2344	-0.43988635E+04	0.24E-01	0.00E+00	0.01
14	0	1448	-0.54812179E+04	0.75E-01	0.00E+00	0.00
15	0	1217	-0.72037317E+04	0.11E-01	0.00E+00	0.00
16	0	1165	-0.85874181E+04	-0.14E-07	0.43E-11	0.00
17	0	1501	-0.97723791E+04	-0.13E-07	0.14E-11	0.00
18	0	1501	-0.10789985E+05	-0.44E-06	0.00E+00	0.00
19	0	1123	-0.11644542E+05	-0.21E-06	0.00E+00	0.00
20	0	850	-0.12317111E+05	-0.96E-07	0.00E+00	0.00
21	0	2234	-0.22360680E+03	0.85E-08	0.80E-09	0.02
22	0	1869	-0.31622791E+03	-0.36E-06	0.37E-06	0.04
23	0	2371	-0.38890875E+03	-0.14E-06	0.53E-07	0.09
24	0	3153	-0.46475801E+03	-0.12E-07	0.75E-08	0.13
25	0	6861	-0.54382902E+03	-0.32E-07	0.28E-10	0.42
26	0	6961	-0.62749502E+03	-0.32E-07	0.10E-10	0.78
27	0	10708	-0.71554175E+03	0.66E-07	0.69E-10	1.76
28	0	15151	-0.80638080E+03	-0.42E-08	0.35E-10	2.62
29	0	19805	-0.90124926E+03	-0.18E-06	0.10E-06	6.02
30	0	20293	-0.99636843E+03	-0.27E-07	0.18E-07	10.37
31	0	22772	-0.10831667E+04	0.32E-06	0.10E-08	11.32
32	0	25115	-0.11635076E+04	0.32E-06	0.51E-09	17.24
33	0	28212	-0.12386485E+04	-0.37E-06	0.34E-09	19.02
34	0	34555	-0.13094846E+04	0.28E-06	0.95E-10	34.15
35	0	43220	-0.13766819E+04	-0.63E-06	0.92E-06	50.19
36	0	45873	-0.13997333E+04	0.28E-01	0.76E-06	51.03
37	0	48206	-0.14797804E+04	0.15E-01	0.45E-08	65.21
38	0	62813	-0.22150059E+04	0.39E-01	0.11E-06	89.33
39	0	75649	-0.29663117E+04	0.25E-01	0.54E-06	144.17
40	0	78536	-0.37405888E+04	0.54E-02	0.16E-06	102.90
41	0	90278	-0.44681105E+04	0.82E-02	0.32E-06	170.03
42	0	101167	-0.51894847E+04	0.64E-02	0.41E-07	146.92
43	0	102971	-0.59231541E+04	0.78E-05	0.90E-07	68.36
44	0	103728	-0.65832193E+04	0.39E-02	0.48E-06	56.85
45	1	104068	-0.72649846E+04	0.23E-02	0.30E-07	29.68
46	1	104292	-0.79276734E+04	0.18E-02	0.80E-06	19.30
47	0	76865	-0.85874181E+04	-0.21E-05	0.16E-07	5.74
48	0	52110	-0.92014325E+04	-0.35E-05	0.17E-07	1.52
49	0	26443	-0.97723810E+04	0.19E-05	0.27E-06	0.32
50	0	15127	-0.10301598E+05	-0.58E-04	0.84E-07	0.10
51	0	11069	-0.10789986E+05	0.13E-05	0.26E-06	0.05

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
52	0	2942	-0.11237843E+05	0.14E-04	0.23E-07	0.01
53	0	1561	-0.11644544E+05	0.39E-04	0.36E-06	0.01
54	0	1497	-0.12007633E+05	0.31E-04	0.59E-06	0.00
55	0	629	-0.12317111E+05	-0.90E-05	0.34E-16	0.00

Tab. A.15: Detailed Results of a Linear Outer Approximation Method for the Shell Test Set

Finally, the detailed results of the NLP-based branch-and-bound method MINLPB4 are shown in Table A.16 for the Shell test set.

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
1	0	4242	-0.16044944E+04	0.35E-05	0.00E+00	0.01
2	0	41131	-0.18000000E+04	-0.41E-11	0.12E-10	0.19
3	0	46820	-0.23082919E+04	0.35E-05	0.33E-10	0.26
4	0	26664	-0.25082919E+04	0.20E-07	0.16E-11	0.17
5	0	36104	-0.26265882E+04	0.45E-05	0.00E+00	0.20
6	0	61224	-0.28044944E+04	0.20E-05	0.35E-10	0.30
7	0	53144	-0.30995309E+04	-0.10E-04	0.15E-10	0.27
8	0	148592	-0.26310453E+04	-0.17E-04	0.10E-11	1.37
9	0	157099	-0.30406261E+04	-0.92E-08	0.28E-11	1.45
10	0	258510	-0.34044944E+04	0.17E-05	0.35E-10	2.25
11	0	11429	-0.15020819E+04	0.74E-07	0.53E-07	0.03
12	0	3151	-0.30418744E+04	-0.14E-06	0.00E+00	0.01
13	0	4678	-0.45049972E+04	-0.49E-07	0.00E+00	0.02
14	0	716	-0.59231534E+04	0.79E-05	0.16E-09	0.00
15	0	2652	-0.72820324E+04	0.80E-07	0.00E+00	0.01
16	0	1285	-0.85729953E+04	0.17E-02	0.00E+00	0.01
17	0	1178	-0.97723788E+04	0.21E-07	0.63E-10	0.01
18	0	1239	-0.10789985E+05	-0.43E-06	0.00E+00	0.00
19	0	945	-0.11644542E+05	-0.18E-06	0.00E+00	0.00
20	0	840	-0.12317111E+05	-0.96E-07	0.00E+00	0.00
21	0	10435	-0.22360680E+03	0.10E-07	0.43E-11	0.05
22	0	13431	-0.31622777E+03	0.11E-06	0.54E-12	0.06
23	0	17380	-0.38890873E+03	-0.76E-07	0.26E-11	0.07
24	0	29307	-0.46475800E+03	-0.34E-08	0.38E-10	0.11
25	0	60881	-0.54382902E+03	-0.32E-07	0.41E-12	0.35
26	0	101764	-0.62749502E+03	-0.32E-07	0.34E-11	0.69
27	0	135235	-0.71554175E+03	0.66E-07	0.92E-12	1.69
28	0	166490	-0.80638080E+03	-0.41E-08	0.33-134	1.90
29	0	194240	-0.90124913E+03	-0.37E-07	0.60E-12	1.61
30	0	93376	-0.99636841E+03	-0.58E-08	0.16E-97	1.16
31	0	112026	-0.10831667E+04	0.32E-06	0.21E-11	0.65
32	0	72788	-0.11635076E+04	0.32E-06	0.67E-12	0.22
33	0	80193	-0.12386485E+04	-0.37E-06	0.24E-12	0.24
34	0	97214	-0.13094846E+04	0.28E-06	0.11E-12	0.28
35	0	168280	-0.13766808E+04	0.15E-06	0.88E-13	0.70

TP	IFAIL	N_f	F	OBJ_ERR	VIOL	TIME
36	0	166996	-0.14407463E+04	-0.23E-06	0.36E-13	0.46
37	0	252654	-0.15020819E+04	0.12E-04	0.12E-30	0.92
38	0	382609	-0.23040725E+04	0.12E-04	0.68E-10	0.96
39	0	592379	-0.30418744E+04	0.84E-05	0.83E-14	1.41
40	0	834812	-0.37609839E+04	0.43E-05	0.58E-13	1.87
41	0	897972	-0.45049972E+04	0.62E-06	0.97E-13	2.08
42	0	914388	-0.52231456E+04	-0.87E-05	0.19E-10	2.15
43	0	797857	-0.59231537E+04	0.78E-05	0.83E-11	1.88
44	0	611168	-0.66090846E+04	0.23E-05	0.35E-10	1.44
45	0	402145	-0.72820326E+04	-0.45E-05	0.49E-11	0.97
46	0	246640	-0.79417410E+04	-0.52E-05	0.92E-11	0.62
47	0	139950	-0.85874181E+04	-0.21E-05	0.23E-10	0.37
48	0	71172	-0.92014325E+04	-0.35E-05	0.73E-12	0.21
49	0	41408	-0.97723791E+04	0.21E-05	0.52E-13	0.13
50	0	26367	-0.10301598E+05	-0.58E-04	0.93E-14	0.08
51	0	23636	-0.10789985E+05	0.14E-05	0.26E-13	0.06
52	0	19521	-0.11237843E+05	0.14E-04	0.64E-11	0.06
53	0	10254	-0.11644542E+05	0.39E-04	0.24E-13	0.04
54	0	6535	-0.12007631E+05	0.31E-04	0.88E-12	0.03
55	0	567	-0.12317111E+05	-0.90E-05	0.34E-16	0.00

Tab. A.16: Detailed Results of MINLPB4 for the Shell Test Set

B. DETAILED MIQP RESULTS

In the sequel we present the detailed results of all 46 MIQP test cases. We compare the following criteria.

Abreviation	Description
TP	Number of the test case.
SETTING	Solver setting, see Table 6.4.
NODES	Number of branch-and-bound nodes.
TIME	Calculation time in seconds.
REDGAP	Gap closed by cutting planes.
CUTS	Number of generated cutting planes.
F	Optimal objective value.
ER	Number of QP failures.
GAP	Optimality gap.

Tab. B.1: Criteria for detailed MIQP Results

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
1	$MIQL_1$	8647	0.123E+01	0.000E+00	0	-0.319588E+04	0	0
1	$MIQL_2$	8647	0.143D+01	0.000D+00	0	-0.319588D+04	0	0
1	$MIQL_3$	8647	0.138E+01	0.000E+00	0	-0.319588E+04	0	0
1	$MIQL_4$	5777	0.135E+01	0.000E+00	0	-0.319588E+04	4	0
1	$MIQL_5$	8319	0.206E+01	0.000E+00	0	-0.319588E+04	0	0
1	$MIQL_6$	8319	0.205E+01	0.000E+00	0	-0.319588E+04	0	0
1	$MIQL_7$	8597	0.202E+01	0.000E+00	0	-0.319588E+04	0	0
1	$MIQL_8$	8659	0.157E+01	0.000E+00	0	-0.319588E+04	0	0
1	SCIP	–	0.493E+01	–	–	-0.319588E+04	0	0
2	$MIQL_1$	79522	0.193E+02	0.000E+00	0	0.799171E+04	0	0
2	$MIQL_2$	79522	0.222E+02	0.000E+00	0	0.799171E+04	0	0
2	$MIQL_3$	79522	0.205E+02	0.000E+00	0	0.799171E+04	0	0
2	$MIQL_4$	79522	0.236E+02	0.000E+00	0	0.799171E+04	0	0
2	$MIQL_5$	70598	0.361E+02	0.000E+00	0	0.799171E+04	0	0
2	$MIQL_6$	70598	0.365E+02	0.000E+00	0	0.799171E+04	0	0
2	$MIQL_7$	79061	0.497E+02	0.000E+00	0	0.799171E+04	0	0
2	$MIQL_8$	134790	0.356E+02	0.000E+00	0	0.799171E+04	1	0
2	SCIP	–	0.363E+01	–	–	0.799171E+04	0	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
3	$MIQL_1$	39878	0.611E+01	0.000E+00	0	0.240435E+04	11	0
3	$MIQL_2$	39878	0.726E+01	0.000E+00	0	0.240435E+04	11	0
3	$MIQL_3$	39878	0.673E+01	0.000E+00	0	0.240435E+04	11	0
3	$MIQL_4$	39881	0.791E+01	0.000E+00	0	0.240435E+04	12	0
3	$MIQL_5$	34811	0.105E+02	0.000E+00	0	0.240435E+04	11	0
3	$MIQL_6$	34811	0.106E+02	0.000E+00	0	0.240435E+04	11	0
3	$MIQL_7$	38799	0.125E+02	0.000E+00	0	0.240435E+04	17	0
3	$MIQL_8$	47603	0.735E+01	0.000E+00	0	0.240435E+04	11	0
3	SCIP	–	0.102E+02	–	–	0.240435E+04	0	0
4	$MIQL_1$	76604	0.168E+02	0.000E+00	0	-0.195681E+00	84	0
4	$MIQL_2$	51987	0.108E+02	0.746E-01	3	-0.195681E+00	48	0
4	$MIQL_3$	59298	0.128E+02	0.487E-01	2	-0.195681E+00	377	0
4	$MIQL_4$	76963	0.216E+02	0.000E+00	0	-0.195681E+00	122	0
4	$MIQL_5$	65888	0.273E+02	0.000E+00	0	-0.195681E+00	117	0
4	$MIQL_6$	43796	0.171E+02	0.487E-01	2	-0.195681E+00	428	0
4	$MIQL_7$	68821	0.301E+02	0.000E+00	0	-0.195681E+00	131	0
4	$MIQL_8$	93480	0.203E+02	0.000E+00	0	-0.195681E+00	106	0
4	SCIP	–	0.258E+01	–	–	-0.195681E+00	0	0
5	$MIQL_1$	106916	0.233E+02	0.000E+00	0	0.170000E+02	16	0
5	$MIQL_2$	83653	0.193E+02	0.110E-01	4	0.170000E+02	3	0
5	$MIQL_3$	84928	0.178E+02	0.110E-01	4	0.170000E+02	24	0
5	$MIQL_4$	106505	0.293E+02	0.000E+00	0	0.170000E+02	10	0
5	$MIQL_5$	80499	0.349E+02	0.000E+00	0	0.170000E+02	3	0
5	$MIQL_6$	64959	0.244E+02	0.110E-01	4	0.170000E+02	28	0
5	$MIQL_7$	95023	0.478E+02	0.000E+00	0	0.170000E+02	3	0
5	$MIQL_8$	150495	0.405E+02	0.000E+00	0	0.170000E+02	2	0
5	SCIP	–	0.282E+01	–	–	0.170000E+02	0	0
6	$MIQL_1$	21468	0.292E+01	0.000E+00	0	0.111722E-02	6	0
6	$MIQL_2$	13404	0.233E+01	0.470E-01	6	0.111722E-02	8	0
6	$MIQL_3$	15543	0.236E+01	0.470E-01	4	0.111722E-02	21	0
6	$MIQL_4$	21596	0.402E+01	0.000E+00	0	0.111722E-02	9	0
6	$MIQL_5$	15818	0.399E+01	0.000E+00	0	0.111722E-02	9	0
6	$MIQL_6$	11714	0.293E+01	0.470E-01	4	0.111722E-02	32	0
6	$MIQL_7$	19045	0.499E+01	0.000E+00	0	0.111722E-02	9	0
6	$MIQL_8$	31723	0.360E+01	0.000E+00	0	0.111722E-02	5	0
6	SCIP	–	0.655E+00	–	–	0.158582E-02	0	0
7	$MIQL_1$	12773	0.151E+01	0.000E+00	0	-0.284760E+00	9	0
7	$MIQL_2$	8514	0.146E+01	0.842E-01	3	-0.284760E+00	14	0
7	$MIQL_3$	9061	0.128E+01	0.234E-01	1	-0.284760E+00	10	0
7	$MIQL_4$	13465	0.225E+01	0.000E+00	0	-0.284760E+00	9	0
7	$MIQL_5$	10022	0.240E+01	0.000E+00	0	-0.284760E+00	6	0
7	$MIQL_6$	7943	0.183E+01	0.234E-01	1	-0.284760E+00	11	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
7	$MIQL_7$	11553	0.292E+01	0.000E+00	0	-0.284760E+00	13	0
7	$MIQL_8$	20305	0.202E+01	0.000E+00	0	-0.284760E+00	13	0
7	SCIP	–	0.508E+00	–	–	-0.284760E+00	0	0
8	$MIQL_1$	27282	0.400E+01	0.000E+00	0	-0.219722E+00	49	0
8	$MIQL_2$	22740	0.392E+01	0.878E-01	2	-0.219722E+00	42	0
8	$MIQL_3$	22696	0.361E+01	0.878E-01	2	-0.219722E+00	64	0
8	$MIQL_4$	27263	0.534E+01	0.000E+00	0	-0.219722E+00	55	0
8	$MIQL_5$	23381	0.642E+01	0.000E+00	0	-0.219722E+00	61	0
8	$MIQL_6$	17392	0.514E+01	0.878E-01	2	-0.219722E+00	77	0
8	$MIQL_7$	24859	0.725E+01	0.000E+00	0	-0.219722E+00	67	0
8	$MIQL_8$	27982	0.397E+01	0.000E+00	0	-0.219722E+00	46	0
8	SCIP	–	0.977E+00	–	–	-0.219722E+00	0	0
9	$MIQL_1$	149440	0.646E+02	0.000E+00	0	-0.348897E+00	198	0
9	$MIQL_2$	111596	0.465E+02	0.745E-01	2	-0.348897E+00	392	0
9	$MIQL_3$	110939	0.434E+02	0.745E-01	2	-0.348897E+00	451	0
9	$MIQL_4$	148234	0.830E+02	0.000E+00	0	-0.348897E+00	260	0
9	$MIQL_5$	96103	0.784E+02	0.000E+00	0	-0.348897E+00	170	0
9	$MIQL_6$	87316	0.612E+02	0.745E-01	2	-0.348897E+00	535	0
9	$MIQL_7$	117467	0.126E+03	0.000E+00	0	-0.348897E+00	286	0
9	$MIQL_8$	173382	0.755E+02	0.000E+00	0	-0.348897E+00	208	0
9	SCIP	–	0.111E+02	–	–	-0.334469E+00	0	0
10	$MIQL_1$	22914	0.381E+01	0.000E+00	0	0.123422E+01	127	0
10	$MIQL_2$	12713	0.358E+01	0.631E-01	6	0.123422E+01	0	0
10	$MIQL_3$	12584	0.289E+01	0.631E-01	5	0.123422E+01	0	0
10	$MIQL_4$	22618	0.552E+01	0.000E+00	0	0.123422E+01	137	0
10	$MIQL_5$	6610	0.210E+01	0.000E+00	0	0.123422E+01	20	0
10	$MIQL_6$	10268	0.382E+01	0.631E-01	5	0.123422E+01	0	0
10	$MIQL_7$	16491	0.658E+01	0.000E+00	0	0.123422E+01	170	0
10	$MIQL_8$	43821	0.601E+01	0.000E+00	0	0.123422E+01	94	0
10	SCIP	–	0.111E+03	–	–	0.123422E+01	0	0
11	$MIQL_1$	30261	0.592E+01	0.000E+00	0	-0.731080E+00	45	0
11	$MIQL_2$	18236	0.494E+01	0.573E-01	4	-0.731080E+00	36	0
11	$MIQL_3$	19870	0.441E+01	0.573E-01	4	-0.731080E+00	45	0
11	$MIQL_4$	30532	0.817E+01	0.000E+00	0	-0.731080E+00	57	0
11	$MIQL_5$	21996	0.868E+01	0.000E+00	0	-0.731080E+00	46	0
11	$MIQL_6$	16351	0.594E+01	0.573E-01	4	-0.731080E+00	52	0
11	$MIQL_7$	26403	0.109E+02	0.000E+00	0	-0.731080E+00	71	0
11	$MIQL_8$	37473	0.673E+01	0.000E+00	0	-0.731080E+00	49	0
11	SCIP	–	0.117E+02	–	–	-0.725985E+00	0	0
12	$MIQL_1$	169398	0.821E+02	0.000E+00	0	0.115009E+00	310	0
12	$MIQL_2$	193766	0.120E+03	0.627E-01	7	0.115009E+00	365	0
12	$MIQL_3$	207939	0.128E+03	0.627E-01	6	0.115009E+00	859	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
12	$MIQL_4$	170839	0.105E+03	0.000E+00	0	0.115009E+00	400	0
12	$MIQL_5$	118490	0.115E+03	0.000E+00	0	0.115009E+00	371	0
12	$MIQL_6$	117410	0.115E+03	0.627E-01	6	0.115009E+00	494	0
12	$MIQL_7$	144237	0.172E+03	0.000E+00	0	0.115009E+00	437	0
12	$MIQL_8$	248790	0.136E+03	0.000E+00	0	0.115009E+00	273	0
12	SCIP	–	0.281E+01	–	–	0.116108E+00	0	0
13	$MIQL_1$	34054	0.734E+01	0.000E+00	0	-0.886997E+00	29	0
13	$MIQL_2$	50693	0.142E+02	0.405E-01	5	-0.886997E+00	89	0
13	$MIQL_3$	53604	0.134E+02	0.203E-01	3	-0.886997E+00	52	0
13	$MIQL_4$	33986	0.971E+01	0.000E+00	0	-0.886997E+00	31	0
13	$MIQL_5$	26627	0.109E+02	0.000E+00	0	-0.886997E+00	51	0
13	$MIQL_6$	40476	0.184E+02	0.203E-01	3	-0.886997E+00	79	0
13	$MIQL_7$	31913	0.135E+02	0.000E+00	0	-0.886997E+00	53	0
13	$MIQL_8$	46892	0.907E+01	0.000E+00	0	-0.886997E+00	45	0
13	SCIP	–	0.132E+02	–	–	-0.886997E+00	0	0
14	$MIQL_1$	10433	0.140E+01	0.000E+00	0	0.313804E-01	4	0
14	$MIQL_2$	11177	0.222E+01	0.635E-01	2	0.313804E-01	4	0
14	$MIQL_3$	11327	0.185E+01	0.635E-01	2	0.313804E-01	166	0
14	$MIQL_4$	10427	0.215E+01	0.000E+00	0	0.313804E-01	9	0
14	$MIQL_5$	6086	0.178E+01	0.000E+00	0	0.313804E-01	11	0
14	$MIQL_6$	6051	0.193E+01	0.635E-01	2	0.313804E-01	221	0
14	$MIQL_7$	7269	0.219E+01	0.000E+00	0	0.313804E-01	15	0
14	$MIQL_8$	16085	0.173E+01	0.000E+00	0	0.313804E-01	3	0
14	SCIP	–	0.156E+01	–	–	0.463081E-01	0	0
15	$MIQL_1$	193056	0.104E+03	0.000E+00	0	0.519714E+01	16	0
15	$MIQL_2$	131785	0.658E+02	0.109E-01	3	0.519714E+01	11	0
15	$MIQL_3$	131576	0.564E+02	0.109E-01	3	0.519714E+01	8	0
15	$MIQL_4$	197309	0.138E+03	0.000E+00	0	0.519714E+01	45	0
15	$MIQL_5$	152190	0.138E+03	0.000E+00	0	0.519714E+01	43	0
15	$MIQL_6$	101550	0.739E+02	0.109E-01	3	0.519714E+01	11	0
15	$MIQL_7$	160425	0.168E+03	0.000E+00	0	0.519714E+01	50	0
15	$MIQL_8$	496389	0.298E+03	0.000E+00	0	0.519714E+01	58	0
15	SCIP	–	0.111E+03	–	–	0.519714E+01	0	0
16	$MIQL_1$	79994	0.192E+02	0.000E+00	0	0.430903E+01	40	0
16	$MIQL_2$	73175	0.206E+02	0.140E-01	2	0.430903E+01	0	0
16	$MIQL_3$	76601	0.190E+02	0.140E-01	2	0.430903E+01	0	0
16	$MIQL_4$	80646	0.294E+02	0.000E+00	0	0.430903E+01	83	0
16	$MIQL_5$	48985	0.220E+02	0.000E+00	0	0.430903E+01	61	0
16	$MIQL_6$	34768	0.176E+02	0.140E-01	2	0.430903E+01	0	0
16	$MIQL_7$	52699	0.242E+02	0.000E+00	0	0.430903E+01	83	0
16	$MIQL_8$	98569	0.224E+02	0.000E+00	0	0.430903E+01	36	0
16	SCIP	–	0.892E+01	–	–	0.430903E+01	0	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
17	$MIQL_1$	18517	0.255E+01	0.000E+00	0	-0.923139E+00	2	0
17	$MIQL_2$	12441	0.268E+01	0.473E-01	3	-0.923139E+00	3	0
17	$MIQL_3$	12808	0.204E+01	0.835E-01	3	-0.923139E+00	7	0
17	$MIQL_4$	18471	0.392E+01	0.000E+00	0	-0.923139E+00	5	0
17	$MIQL_5$	9674	0.297E+01	0.000E+00	0	-0.923139E+00	3	0
17	$MIQL_6$	6977	0.216E+01	0.835E-01	3	-0.923139E+00	10	0
17	$MIQL_7$	11517	0.370E+01	0.000E+00	0	-0.923139E+00	4	0
17	$MIQL_8$	28850	0.327E+01	0.000E+00	0	-0.923139E+00	2	0
17	SCIP	–	0.521E+01	–	–	-0.907496E+00	0	0
18	$MIQL_1$	11780	0.158E+01	0.000E+00	0	0.000000E+00	2	0
18	$MIQL_2$	7635	0.208E+01	0.400E-01	4	0.000000E+00	2	0
18	$MIQL_3$	6401	0.142E+01	0.418E-01	2	0.000000E+00	7	0
18	$MIQL_4$	7791	0.171E+01	0.000E+00	0	0.000000E+00	3	0
18	$MIQL_5$	7435	0.216E+01	0.000E+00	0	0.000000E+00	3	0
18	$MIQL_6$	6163	0.185E+01	0.418E-01	2	0.000000E+00	9	0
18	$MIQL_7$	7723	0.227E+01	0.000E+00	0	0.000000E+00	3	0
18	$MIQL_8$	7757	0.124E+01	0.000E+00	0	0.000000E+00	1	0
18	SCIP	–	0.373E+00	–	–	0.000000E+00	0	0
19	$MIQL_1$	189613	0.112E+03	0.000E+00	0	0.146042E+02	7	0
19	$MIQL_2$	161836	0.111E+03	0.734E-03	1	0.146042E+02	3	0
19	$MIQL_3$	189613	0.116E+03	0.000E+00	0	0.146042E+02	7	0
19	$MIQL_4$	189255	0.156E+03	0.000E+00	0	0.146042E+02	40	0
19	$MIQL_5$	100118	0.116E+03	0.000E+00	0	0.146042E+02	1	0
19	$MIQL_6$	100118	0.117E+03	0.000E+00	0	0.146042E+02	1	0
19	$MIQL_7$	124029	0.177E+03	0.000E+00	0	0.146042E+02	1	0
19	$MIQL_8$	629914	0.327E+03	0.000E+00	0	0.146042E+02	79	0
19	SCIP	–	0.510E+03	–	–	0.146042E+02	0	0
20	$MIQL_1$	136727	0.629E+02	0.000E+00	0	0.191010E+01	218	0
20	$MIQL_2$	137257	0.806E+02	0.473E-03	1	0.191010E+01	216	0
20	$MIQL_3$	136727	0.671E+02	0.000E+00	0	0.191010E+01	218	0
20	$MIQL_4$	149690	0.109E+03	0.000E+00	0	0.191010E+01	362	0
20	$MIQL_5$	65471	0.646E+02	0.000E+00	0	0.191010E+01	281	0
20	$MIQL_6$	65471	0.650E+02	0.000E+00	0	0.191010E+01	281	0
20	$MIQL_7$	82859	0.988E+02	0.000E+00	0	0.191010E+01	359	0
20	$MIQL_8$	533183	0.217E+03	0.000E+00	0	0.191010E+01	560	0
20	SCIP	–	0.234E+03	–	–	0.191010E+01	0	0
21	$MIQL_1$	6216	0.134E+01	0.000E+00	0	-0.499951E+00	15	0
21	$MIQL_2$	6216	0.790E+01	0.000E+00	0	-0.499951E+00	15	0
21	$MIQL_3$	6216	0.251E+01	0.000E+00	0	-0.499951E+00	15	0
21	$MIQL_4$	6239	0.250E+01	0.000E+00	0	-0.499951E+00	15	0
21	$MIQL_5$	3083	0.172E+01	0.000E+00	0	-0.499951E+00	21	0
21	$MIQL_6$	3083	0.177E+01	0.000E+00	0	-0.499951E+00	21	0
21	$MIQL_7$	3805	0.218E+01	0.000E+00	0	-0.499951E+00	14	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
21	$MIQL_8$	24163	0.285E+01	0.000E+00	0	-0.499951E+00	44	0
21	SCIP	–	0.235E+02	–	–	-0.499951E+00	0	0
22	$MIQL_1$	44888	0.168E+02	0.000E+00	0	0.271000E+02	6	0
22	$MIQL_2$	18529	0.120E+02	0.147E+00	3	0.271000E+02	0	0
22	$MIQL_3$	21121	0.840E+01	0.147E+00	3	0.271000E+02	3	0
22	$MIQL_4$	36946	0.210E+02	0.000E+00	0	0.271000E+02	4	0
22	$MIQL_5$	28134	0.217E+02	0.000E+00	0	0.271000E+02	3	0
22	$MIQL_6$	12457	0.885E+01	0.147E+00	3	0.271000E+02	0	0
22	$MIQL_7$	36761	0.280E+02	0.000E+00	0	0.271000E+02	5	0
22	$MIQL_8$	47171	0.169E+02	0.000E+00	0	0.271000E+02	0	0
22	SCIP	–	0.294E+01	–	–	0.271000E+02	0	0
23	$MIQL_1$	38338	0.617E+01	0.000E+00	0	0.135918E+01	4	0
23	$MIQL_2$	31398	0.575E+01	0.474E-01	3	0.135918E+01	5	0
23	$MIQL_3$	32018	0.553E+01	0.460E-01	2	0.135918E+01	24	0
23	$MIQL_4$	38223	0.772E+01	0.000E+00	0	0.135918E+01	4	0
23	$MIQL_5$	30621	0.880E+01	0.000E+00	0	0.135918E+01	6	0
23	$MIQL_6$	26636	0.746E+01	0.460E-01	2	0.135918E+01	20	0
23	$MIQL_7$	36507	0.105E+02	0.000E+00	0	0.135918E+01	10	0
23	$MIQL_8$	42078	0.632E+01	0.000E+00	0	0.135918E+01	2	0
23	SCIP	–	0.189E+01	–	–	0.135918E+01	0	0
24	$MIQL_1$	26573	0.752E+01	0.000E+00	0	-0.842309E+00	9	0
24	$MIQL_2$	29294	0.122E+02	0.170E-01	7	-0.842309E+00	4	0
24	$MIQL_3$	32584	0.110E+02	0.106E-01	5	-0.842309E+00	99	0
24	$MIQL_4$	26736	0.109E+02	0.000E+00	0	-0.842309E+00	14	0
24	$MIQL_5$	17331	0.106E+02	0.000E+00	0	-0.842309E+00	8	0
24	$MIQL_6$	21184	0.142E+02	0.106E-01	5	-0.842309E+00	85	0
24	$MIQL_7$	19413	0.122E+02	0.000E+00	0	-0.842309E+00	13	0
24	$MIQL_8$	32052	0.809E+01	0.000E+00	0	-0.842309E+00	7	0
24	SCIP	–	0.181E+01	–	–	-0.828992E+00	0	0
25	$MIQL_1$	289359	0.185E+03	0.000E+00	0	-0.462977E+00	247	0
25	$MIQL_2$	480975	0.499E+03	0.120E+00	6	-0.462977E+00	1000	0
25	$MIQL_3$	503419	0.480E+03	0.120E+00	4	-0.462977E+00	1000	0
25	$MIQL_4$	293059	0.293E+03	0.000E+00	0	-0.462977E+00	823	0
25	$MIQL_5$	118458	0.200E+03	0.000E+00	0	-0.462977E+00	283	0
25	$MIQL_6$	199008	0.474E+03	0.120E+00	4	-0.462977E+00	1000	0
25	$MIQL_7$	130509	0.252E+03	0.000E+00	0	-0.462977E+00	472	0
25	$MIQL_8$	929642	0.509E+03	0.000E+00	0	-0.462977E+00	273	0
25	SCIP	–	0.703E+01	–	–	-0.462977E+00	0	0
26	$MIQL_1$	24673	0.102E+02	0.000E+00	0	0.244210E+01	0	0
26	$MIQL_2$	21545	0.203E+02	0.939E-01	2	0.244210E+01	500	0
26	$MIQL_3$	1117	0.177E+01	0.188E+00	4	0.244210E+01	83	0
26	$MIQL_4$	24917	0.140E+02	0.000E+00	0	0.244210E+01	0	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
26	MIQL$_5$	20020	0.160E+02	0.000E+00	0	0.244210E+01	0	0
26	MIQL$_6$	675	0.572E+00	0.188E+00	4	0.244210E+01	101	0
26	MIQL$_7$	23227	0.192E+02	0.000E+00	0	0.244210E+01	0	0
26	MIQL$_8$	155241	0.356E+02	0.000E+00	0	0.244210E+01	9	0
26	SCIP	–	0.960E+02	–	–	0.244210E+01	0	0
27	MIQL$_1$	53338	0.270E+02	0.000E+00	0	0.103579E+02	3	0
27	MIQL$_2$	53338	0.349E+02	0.000E+00	0	0.103579E+02	3	0
27	MIQL$_3$	53338	0.297E+02	0.000E+00	0	0.103579E+02	3	0
27	MIQL$_4$	45038	0.293E+02	0.000E+00	0	0.103579E+02	7	0
27	MIQL$_5$	46024	0.454E+02	0.000E+00	0	0.103579E+02	9	0
27	MIQL$_6$	46024	0.455E+02	0.000E+00	0	0.103579E+02	9	0
27	MIQL$_7$	44269	0.449E+02	0.000E+00	0	0.103579E+02	10	0
27	MIQL$_8$	371341	0.108E+03	0.000E+00	0	0.103579E+02	72	0
27	SCIP	–	0.511E+02	–	–	0.103579E+02	0	0
28	MIQL$_1$	9714	0.768E+01	0.000E+00	0	0.218394E+01	69	0
28	MIQL$_2$	9862	0.106E+02	0.684E-01	3	0.218394E+01	55	0
28	MIQL$_3$	29134	0.237E+02	0.168E+00	6	0.218394E+01	117	0
28	MIQL$_4$	10016	0.122E+02	0.000E+00	0	0.218394E+01	131	0
28	MIQL$_5$	7048	0.124E+02	0.000E+00	0	0.218394E+01	37	0
28	MIQL$_6$	19395	0.346E+02	0.168E+00	6	0.218394E+01	102	0
28	MIQL$_7$	7733	0.140E+02	0.000E+00	0	0.218394E+01	126	0
28	MIQL$_8$	22511	0.104E+02	0.000E+00	0	0.218394E+01	94	0
28	SCIP	–	0.152E+01	–	–	0.218394E+01	0	0
29	MIQL$_1$	278355	0.243E+03	0.000E+00	0	0.761000E+02	0	0
29	MIQL$_2$	445727	0.643E+03	0.340E+00	4	0.761000E+02	3	0
29	MIQL$_3$	603063	0.936E+03	0.318E+00	4	0.761000E+02	190	0
29	MIQL$_4$	273682	0.310E+03	0.000E+00	0	0.761000E+02	0	0
29	MIQL$_5$	208926	0.384E+03	0.000E+00	0	0.761000E+02	0	0
29	MIQL$_6$	450251	0.150E+04	0.318E+00	4	0.761000E+02	141	0
29	MIQL$_7$	241681	0.496E+03	0.000E+00	0	0.761000E+02	0	0
29	MIQL$_8$	289760	0.262E+03	0.000E+00	0	0.761000E+02	0	0
29	SCIP	–	0.390E+00	–	–	0.761000E+02	0	0
30	MIQL$_1$	45066	0.210E+02	0.000E+00	0	-0.184565E+00	0	0
30	MIQL$_2$	45066	0.245E+02	0.000E+00	0	-0.184565E+00	0	0
30	MIQL$_3$	45066	0.217E+02	0.000E+00	0	-0.184565E+00	0	0
30	MIQL$_4$	45066	0.292E+02	0.000E+00	0	-0.184565E+00	0	0
30	MIQL$_5$	33226	0.301E+02	0.000E+00	0	-0.184565E+00	0	0
30	MIQL$_6$	33226	0.302E+02	0.000E+00	0	-0.184565E+00	0	0
30	MIQL$_7$	44069	0.416E+02	0.000E+00	0	-0.184565E+00	0	0
30	MIQL$_8$	58852	0.231E+02	0.000E+00	0	-0.184565E+00	0	0
30	SCIP	–	0.102E+04	–	–	-0.176184E+00	0	26.42
31	MIQL$_1$	43816	0.205E+02	0.000E+00	0	-0.895764E-01	0	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
31	$MIQL_2$	43816	0.236E+02	0.000E+00	0	-0.895764E-01	0	0
31	$MIQL_3$	43816	0.211E+02	0.000E+00	0	-0.895764E-01	0	0
31	$MIQL_4$	43816	0.285E+02	0.000E+00	0	-0.895764E-01	0	0
31	$MIQL_5$	34040	0.314E+02	0.000E+00	0	-0.895764E-01	0	0
31	$MIQL_6$	34040	0.315E+02	0.000E+00	0	-0.895764E-01	0	0
31	$MIQL_7$	42523	0.399E+02	0.000E+00	0	-0.895764E-01	0	0
31	$MIQL_8$	51287	0.215E+02	0.000E+00	0	-0.895764E-01	0	0
31	SCIP	–	0.102E+04	–	–	-0.760732E-01	0	48.24
32	$MIQL_1$	18096	0.642E+01	0.000E+00	0	-0.768226E-01	0	0
32	$MIQL_2$	17369	0.863E+01	0.000E+00	0	-0.768226E-01	0	0
32	$MIQL_3$	18096	0.677E+01	0.000E+00	0	-0.768226E-01	0	0
32	$MIQL_4$	17369	0.876E+01	0.000E+00	0	-0.768226E-01	0	0
32	$MIQL_5$	13052	0.911E+01	0.000E+00	0	-0.768226E-01	0	0
32	$MIQL_6$	13052	0.887E+01	0.000E+00	0	-0.768226E-01	0	0
32	$MIQL_7$	17369	0.120E+02	0.000E+00	0	-0.768226E-01	0	0
32	$MIQL_8$	17369	0.635E+01	0.000E+00	0	-0.768226E-01	0	0
32	SCIP	–	0.103E+04	–	–	-0.639556E-01	0	54.91
33	$MIQL_1$	23288	0.854E+01	0.000E+00	0	-0.561962E-01	0	0
33	$MIQL_2$	23288	0.109E+02	0.000E+00	0	-0.561962E-01	0	0
33	$MIQL_3$	23288	0.894E+01	0.000E+00	0	-0.561962E-01	0	0
33	$MIQL_4$	23288	0.125E+02	0.000E+00	0	-0.561962E-01	0	0
33	$MIQL_5$	16644	0.123E+02	0.000E+00	0	-0.561962E-01	0	0
33	$MIQL_6$	16644	0.124E+02	0.000E+00	0	-0.561962E-01	0	0
33	$MIQL_7$	21645	0.168E+02	0.000E+00	0	-0.561962E-01	0	0
33	$MIQL_8$	26311	0.892E+01	0.000E+00	0	-0.561962E-01	0	0
33	SCIP	–	0.103E+04	–	–	-0.407432E-01	0	60.06
34	$MIQL_1$	43946	0.194E+02	0.000E+00	0	-0.285981E+00	0	0
34	$MIQL_2$	43946	0.226E+02	0.000E+00	0	-0.285981E+00	0	0
34	$MIQL_3$	43946	0.201E+02	0.000E+00	0	-0.285981E+00	0	0
34	$MIQL_4$	43946	0.284E+02	0.000E+00	0	-0.285981E+00	0	0
34	$MIQL_5$	29431	0.272E+02	0.000E+00	0	-0.285981E+00	0	0
34	$MIQL_6$	29431	0.273E+02	0.000E+00	0	-0.285981E+00	0	0
34	$MIQL_7$	39077	0.397E+02	0.000E+00	0	-0.285981E+00	0	0
34	$MIQL_8$	92077	0.282E+02	0.000E+00	0	-0.285981E+00	0	0
34	SCIP	–	0.102E+04	–	–	-0.271812E+00	0	20.96
35	$MIQL_1$	20345	0.756E+01	0.000E+00	0	-0.219086E+00	0	0
35	$MIQL_2$	20345	0.962E+01	0.000E+00	0	-0.219086E+00	0	0
35	$MIQL_3$	20345	0.797E+01	0.000E+00	0	-0.219086E+00	0	0
35	$MIQL_4$	20345	0.112E+02	0.000E+00	0	-0.219086E+00	0	0
35	$MIQL_5$	14228	0.112E+02	0.000E+00	0	-0.219086E+00	0	0
35	$MIQL_6$	14228	0.112E+02	0.000E+00	0	-0.219086E+00	0	0
35	$MIQL_7$	18039	0.149E+02	0.000E+00	0	-0.219086E+00	0	0
35	$MIQL_8$	38892	0.933E+01	0.000E+00	0	-0.219086E+00	0	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
35	SCIP	–	0.102E+04	–	–	-0.211989E+00	0	22.99
36	$MIQL_1$	38494	0.153E+02	0.000E+00	0	-0.253280E+00	0	0
36	$MIQL_2$	38494	0.181E+02	0.000E+00	0	-0.253280E+00	0	0
36	$MIQL_3$	38494	0.158E+02	0.000E+00	0	-0.253280E+00	0	0
36	$MIQL_4$	38494	0.233E+02	0.000E+00	0	-0.253280E+00	0	0
36	$MIQL_5$	22923	0.199E+02	0.000E+00	0	-0.253280E+00	0	0
36	$MIQL_6$	22923	0.200E+02	0.000E+00	0	-0.253280E+00	0	0
36	$MIQL_7$	31583	0.295E+02	0.000E+00	0	-0.253280E+00	0	0
36	$MIQL_8$	49315	0.164E+02	0.000E+00	0	-0.253280E+00	0	0
36	SCIP	–	0.102E+04	–	–	-0.243009E+00	0	21.03
37	$MIQL_1$	47912	0.220E+02	0.000E+00	0	-0.169228E+00	0	0
37	$MIQL_2$	47417	0.250E+02	0.000E+00	0	-0.169228E+00	0	0
37	$MIQL_3$	47912	0.226E+02	0.000E+00	0	-0.169228E+00	0	0
37	$MIQL_4$	47417	0.302E+02	0.000E+00	0	-0.169228E+00	0	0
37	$MIQL_5$	38230	0.344E+02	0.000E+00	0	-0.169228E+00	0	0
37	$MIQL_6$	38230	0.316E+02	0.000E+00	0	-0.169228E+00	0	0
37	$MIQL_7$	47417	0.391E+02	0.000E+00	0	-0.169228E+00	0	0
37	$MIQL_8$	47417	0.218E+02	0.000E+00	0	-0.169228E+00	0	0
37	SCIP	–	0.102E+04	–	–	-0.155441E+00	0	30.04
38	$MIQL_1$	42386	0.183E+02	0.000E+00	0	-0.262034E+00	0	0
38	$MIQL_2$	42386	0.212E+02	0.000E+00	0	-0.262034E+00	0	0
38	$MIQL_3$	42386	0.188E+02	0.000E+00	0	-0.262034E+00	0	0
38	$MIQL_4$	42386	0.256E+02	0.000E+00	0	-0.262034E+00	0	0
38	$MIQL_5$	29896	0.251E+02	0.000E+00	0	-0.262034E+00	0	0
38	$MIQL_6$	29896	0.252E+02	0.000E+00	0	-0.262034E+00	0	0
38	$MIQL_7$	40813	0.375E+02	0.000E+00	0	-0.262034E+00	0	0
38	$MIQL_8$	58728	0.208E+02	0.000E+00	0	-0.262034E+00	0	0
38	SCIP	–	0.102E+04	–	–	-0.252765E+00	0	21.92
39	$MIQL_1$	43937	0.198E+02	0.000E+00	0	-0.124097E+00	0	0
39	$MIQL_2$	43937	0.231E+02	0.000E+00	0	-0.124097E+00	0	0
39	$MIQL_3$	43937	0.204E+02	0.000E+00	0	-0.124097E+00	0	0
39	$MIQL_4$	43937	0.275E+02	0.000E+00	0	-0.124097E+00	0	0
39	$MIQL_5$	31482	0.283E+02	0.000E+00	0	-0.124097E+00	0	0
39	$MIQL_6$	31482	0.284E+02	0.000E+00	0	-0.124097E+00	0	0
39	$MIQL_7$	42297	0.398E+02	0.000E+00	0	-0.124097E+00	0	0
39	$MIQL_8$	50707	0.206E+02	0.000E+00	0	-0.124097E+00	0	0
39	SCIP	–	0.102E+04	–	–	-0.121339E+00	0	32.45
40	$MIQL_1$	22439	0.620E+01	0.000E+00	0	-0.318334E+00	0	0
40	$MIQL_2$	20299	0.802E+01	0.000E+00	0	-0.318334E+00	0	0
40	$MIQL_3$	22439	0.655E+01	0.000E+00	0	-0.318334E+00	0	0
40	$MIQL_4$	20299	0.827E+01	0.000E+00	0	-0.318334E+00	0	0
40	$MIQL_5$	11932	0.632E+01	0.000E+00	0	-0.318334E+00	0	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
40	$MIQL_6$	11932	0.626E+01	0.000E+00	0	-0.318334E+00	0	0
40	$MIQL_7$	20299	0.107E+02	0.000E+00	0	-0.318334E+00	0	0
40	$MIQL_8$	20299	0.583E+01	0.000E+00	0	-0.318334E+00	0	0
40	SCIP	–	0.102E+04	–	–	-0.306354E+00	0	17.77
41	$MIQL_1$	93738	0.366E+02	0.000E+00	0	-0.419773E+00	38	0
41	$MIQL_2$	41630	0.168E+02	0.319E-01	7	-0.419773E+00	48	0
41	$MIQL_3$	41842	0.149E+02	0.287E-01	6	-0.419773E+00	54	0
41	$MIQL_4$	93865	0.502E+02	0.000E+00	0	-0.419773E+00	59	0
41	$MIQL_5$	78250	0.597E+02	0.000E+00	0	-0.419773E+00	73	0
41	$MIQL_6$	36889	0.234E+02	0.287E-01	6	-0.419773E+00	100	0
41	$MIQL_7$	81003	0.681E+02	0.000E+00	0	-0.419773E+00	78	0
41	$MIQL_8$	141006	0.535E+02	0.000E+00	0	-0.419773E+00	62	0
41	SCIP	–	0.166E+01	–	–	-0.419773E+00	0	0
42	$MIQL_1$	222092	0.157E+03	0.000E+00	0	0.000000E+00	368	0
42	$MIQL_2$	319065	0.344E+03	0.123E-01	2	0.000000E+00	304	0
42	$MIQL_3$	346281	0.342E+03	0.216E-01	2	0.000000E+00	323	0
42	$MIQL_4$	209647	0.182E+03	0.000E+00	0	0.000000E+00	877	0
42	$MIQL_5$	205734	0.263E+03	0.000E+00	0	0.000000E+00	1000	0
42	$MIQL_6$	327257	0.595E+03	0.216E-01	2	0.000000E+00	701	0
42	$MIQL_7$	210127	0.276E+03	0.000E+00	0	0.000000E+00	1000	0
42	$MIQL_8$	207407	0.138E+03	0.000E+00	0	0.000000E+00	1000	0
42	SCIP	–	0.300E+01	–	–	0.000000E+00	0	0
43	$MIQL_1$	33478	0.135E+02	0.000E+00	0	-0.865117E-01	0	0
43	$MIQL_2$	33478	0.163E+02	0.000E+00	0	-0.865117E-01	0	0
43	$MIQL_3$	33478	0.140E+02	0.000E+00	0	-0.865117E-01	0	0
43	$MIQL_4$	33478	0.193E+02	0.000E+00	0	-0.865117E-01	0	0
43	$MIQL_5$	23953	0.197E+02	0.000E+00	0	-0.865117E-01	0	0
43	$MIQL_6$	23953	0.198E+02	0.000E+00	0	-0.865117E-01	0	0
43	$MIQL_7$	31019	0.273E+02	0.000E+00	0	-0.865117E-01	0	0
43	$MIQL_8$	31989	0.133E+02	0.000E+00	0	-0.865117E-01	0	0
43	SCIP	–	0.103E+04	–	–	-0.743918E-01	0	48.44
44	$MIQL_1$	31389	0.106E+02	0.000E+00	0	-0.588872E-01	0	0
44	$MIQL_2$	31389	0.135E+02	0.000E+00	0	-0.588872E-01	0	0
44	$MIQL_3$	31389	0.112E+02	0.000E+00	0	-0.588872E-01	0	0
44	$MIQL_4$	31389	0.171E+02	0.000E+00	0	-0.588872E-01	0	0
44	$MIQL_5$	18387	0.141E+02	0.000E+00	0	-0.588872E-01	0	0
44	$MIQL_6$	18387	0.142E+02	0.000E+00	0	-0.588872E-01	0	0
44	$MIQL_7$	24403	0.199E+02	0.000E+00	0	-0.588872E-01	0	0
44	$MIQL_8$	31580	0.106E+02	0.000E+00	0	-0.588872E-01	0	0
44	SCIP	–	0.102E+04	–	–	-0.515129E-01	0	55.32
45	$MIQL_1$	37386	0.143E+02	0.000E+00	0	-0.431877E-01	0	0
45	$MIQL_2$	37386	0.175E+02	0.000E+00	0	-0.431877E-01	0	0

TP	SETTING	NODES	TIME	REDGAP	CUTS	F	ER	GAP
45	$MIQL_3$	37386	0.150E+02	0.000E+00	0	-0.431877E-01	0	0
45	$MIQL_4$	37386	0.195E+02	0.000E+00	0	-0.431877E-01	0	0
45	$MIQL_5$	27111	0.201E+02	0.000E+00	0	-0.431877E-01	0	0
45	$MIQL_6$	27111	0.201E+02	0.000E+00	0	-0.431877E-01	0	0
45	$MIQL_7$	36659	0.292E+02	0.000E+00	0	-0.431877E-01	0	0
45	$MIQL_8$	44390	0.152E+02	0.000E+00	0	-0.431877E-01	0	0
45	SCIP	–	0.103E+04	–	–	-0.318330E-01	0	68.58
46	$MIQL_1$	38567	0.153E+02	0.000E+00	0	-0.953241E-01	0	0
46	$MIQL_2$	38567	0.187E+02	0.000E+00	0	-0.953241E-01	0	0
46	$MIQL_3$	38567	0.159E+02	0.000E+00	0	-0.953241E-01	0	0
46	$MIQL_4$	38567	0.213E+02	0.000E+00	0	-0.953241E-01	0	0
46	$MIQL_5$	28295	0.218E+02	0.000E+00	0	-0.953241E-01	0	0
46	$MIQL_6$	28295	0.219E+02	0.000E+00	0	-0.953241E-01	0	0
46	$MIQL_7$	37801	0.311E+02	0.000E+00	0	-0.953241E-01	0	0
46	$MIQL_8$	48813	0.168E+02	0.000E+00	0	-0.953241E-01	0	0
46	SCIP	–	0.102E+04	–	–	-0.791613E-01	0	50.82

Tab. B.2: Detailed MIQP Results

BIBLIOGRAPHY

[1] Abhishek K., Leyffer S., Linderoth J. T. (2010): *FilMINT: An Outer Approximation-Based Solver for Convex Mixed-Integer Nonlinear Programs*, Informs Journal on Computing, Vol. 22, No. 4, Fall 2010, pp. 555-567

[2] Achterberg T. (2004): *SCIP - a framework to integrate constraint and mixed-integer programming*, Technical Report 04-19, Zuse Institute Berlin, Germany

[3] Achterberg T., Koch T., Martin A. (2005): *Branching rules revisited*, Operations Research Letters, Vol. 33, 42–54

[4] Achterberg T. (2009): *Constraint Integer Programming*, Dissertation, Technical university Berlin

[5] Asaadi J. (1973): *A computational comparison of some non-linear programs*, Mathematical Programming, Vol. 4, 144–154

[6] Armijo L. (1966): *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific Journal of Mathematics, Vol. 16, 1–3

[7] Audet C., Dennis J.E. (2001): *Pattern search algorithm for mixed variable programming*, SIAM Journal on Optimization, Vol. 11, 573–594

[8] Ayatollahi S., Narimani M., Moshfeghiam M. (2004): *Intermittent gas lift in Aghjari Oil 488 Field, a mathematical study*, Journal of Petroleum Science and Engineering, Vol. 42, 245-255

[9] Balas E. (1971): *Intersection cuts - a new type of cutting planes for integer programming*, Operations Research, Vol. 19, 19-39

[10] Balas E. (1975): *Disjunctive programming - cutting planes from logical conditions*, Nonlinear Programming, Vol. 2, 330-382

[11] Balas E. (1979): *Disjunctive programming*, Annals of Discrete Mathematics, Vol. 5, 3-51

[12] Balas E. (1998): *Disjunctive programming: Properties of the convex hull of feasible points*, Discrete Applied Mathematics, Vol. 89, 1-44

[13] Balas E., Ceria S., Cornuejols G. (1993): *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming, Vol. 58, 295-324

[14] Balas E., Jeroslow R. (1980): *Strengthening cuts for mixed-integer programs*, European Journal of Operations Research, Vol. 4, 224-234

[15] Balas E., Perregaard M. (2001): *Generating cuts from multiple-term disjunctions*, Lecture Notes in Computer Science, Vol. 2081, 348-360

[16] Balas E., Perregaard M. (2002): *Lift-and-project for mixed 0-1 programming: recent progress*, Discrete Applied Mathematics, Vol. 123, 129-154

[17] Balas E., Perregaard M. (2003): *A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed-integer gomory cuts*, Mathematical Programming, Vol. 94, 221-2245

[18] Balas E., Bonami P. (2008): *New Variants of Lift-and-Project Cut Generation from the LP Tableau: Open Source Implementation and Testing* IPCO 2007, 89-103

[19] Barton P., Selot A. (2007): *A production allocation framework for natural gas production systems*, Computer Aided Chemical Engineering, 24, 539-544

[20] Beling P.A., Megiddo N. (1998): *Using fast matrix multiplication to find basic solutions*, Theoretical Computer Science, Vol. 205, 307-316

[21] Belotti P., Lee J., Liberti L., Margot F., Wächter A. (2009): *Branching and bounds tightening techniques for non-convex MINLP*, Optimization Methods and Software, Vol. 24, 597–634

[22] Benders J. F. (1962): *Partitioning procedures for solving mixed variable programming problems*, Numerische Mathematik, 4, pp. 238-252

[23] Berthold T. (2007): *Heuristics of the Branch-Cut-and-Price-Framework SCIP*, ZIB-Report 07-30, Zuse Institute Berlin

[24] Berthold T., Heinz S., Vigerske S. (2009): *Extending a CIP framework to solve MIQCPs*, ZIB-Report 09-23, Zuse Institute Berlin

[25] Bienstock D. (1994): *Computational study of a family of mixed-integer quadratic programming problems*, Columbia University New York, USA

[26] Bixby R.E., Fenelon M., Gu Z., Rothberg E., and Wunderling R. (2004): *Mixed integer programming: A progress report*, in Martin Grötschel (ed.), *The sharpest cut: The impact of Manfred Padberg and his work*, MPS-SIAM Series on Optimization, Vol. 4

[27] Bonami P., Biegler L. T., Conn A. R., Cornuejols G., Grossmann I. E., Laird C. D., Lee J., Lodi A., Margot F., Sawaya N., Wächter A. (2008): *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, 5

[28] Bonami P., Kilinc M., Linderoth J. (2011): *Alogrithms and software for convex mixed integer nonlinear programs* IMA Volumes, to appear

[29] Borchers B., Mitchell J.E. (1994): *An improved branch-and-bound algorithm for mixed-integer nonlinear programming,* Computers and Operations Research, Vol. 21, No. 4, 359-367

[30] van de Braak G. (2001): *Das Verfahren MISQP zur gemischt ganzzahligen nichtlinearen Programmierung für den Entwurf elektronischer Bauteile*, Diploma Thesis, Department of Numerical and Instrumental Mathematics, University of Münster, Germany

[31] Brooke A., Kendrick D., Meeraus A. (1988): *GAMS: A User's Guide*, The ScientificS

[32] Buchheim C., Caprara A., Lodi A. (2012): *An effective branch-and-bound algorithm for convex quadratic integer programming*, Mathematical Programming, Volume 135, Issue 1-2, pp 369-395

[33] Bünner M.J., Schittkowski K., van de Braak G. (2004): *Optimal design of electronic components by mixed-integer nonlinear programming*, Optimization and Engineering, Vol. 5, 271-294

[34] Bussieck M.R., Drud A.S., Meeraus A. (2007): *MINLPLib - A collection of test models for mixed-integer nonlinear programming*, GAMS Development Corp., Washington D.C., USA

[35] Byrd R.H., Schnabel R.B., Shultz G.A. (1987): *A trust region algorithm for nonlinearly constrained optimization*, SIAM Journal on Numerical Analysis, Vol. 24,erical Analysis, Vol. 24, 1152–1170

[36] Cha J.Z., Mayne R.W. (1989): *Optimization with discrete variables via recursive quadratic programming: Part 2 - algorithms and results*, Transactions of the ASME, Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 111, 130–136

[37] Ceria S., Pataki G. (1998): *Solving integer and disjunctive programs by lift-and-project*, Lecture Notes in Computer Science, Vol. 1412, 271-283

[38] Ceria S., Soares J. (1997): *Disjunctive cut generation for mixed 0-1 programs: duality and lifting*, GSB, Columbia University New York, USA

[39] Chamberlain R.M., Lemarechal C., Pedersen H.C., Powell M.J.D. (1982): *The watchdog technique for forcing convergence in algorithms for constrained optimization*, Mathematical Programming Study, Vol. 16, 1–17

[40] Cornuejols G. (2008): *Valid inequalities for mixed integer linear programming*, Mathematical Programming, Series B, Vol. 112, 3-44

[41] CPLEX, IBM: http://ibm.com/software/integration/optimization/cplex.

[42] Dakin R.J. (1965): *A tree search algorithm for mixed-integer programming problems*, Computer Journal, Vol. 8, 250-255

[43] Duran M., Grossmann I.E. (1986): *An outer-approximation algorithm for a class of Mixed Integer Nonlinear Programs*, Mathematical Programming, Vol. 36, 307-339

[44] Eldred M.S., Giunta A.A.,van Bloemen Waanders B.G., Wojtkiewicz S.F., Hart W.E., Alleva M.D. (2002): *DAKOTA: A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis, Version 3.1 Users Manuel*, Report SAND20001-3796, Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185-0847

[45] Exler O., Schittkowski K. (2007): *A trust region SQP algorithm for mixed-integer nonlinear programming*, Optimization Letters, Vol. 1, No. 3, 269-280

[46] Exler O., Lehmann T., Schittkowski K. (2009): *MISQPN : A Fortran subroutine for mixed-integer nonlinear optimization by outer approximation supported by mixed-integer search steps - user's guide, version 1.0*, Report, Department of Computer Science, University of Bayreuth

[47] Exler O., Lehmann T., Schittkowski K. (2011): *A Comparative Study of SQP-Type Algorithms for Nonlinear and Nonconvex Mixed-Integer Optimization*, submitted to Mathematical Programming Computation

[48] Exler O., Lehmann T., Schittkowski K. (2011): *MISQP: A Fortran subroutine of a trust region SQP algorithm for mixed-integer nonlinear programming - user's guide*, Report, Department of Computer Science, University of Bayreuth

[49] Fletcher R. (1982): *Second order correction for nondifferentiable optimization*, in: Watson G.A., ed., Numerical Analysis, Springer, Berlin, pp. 85–115

[50] Fletcher R., Leyffer S. (1994): *Solving mixed-integer nonlinear programs by outer approximation*, Mathematical Programming, Vol. 66, 327-349

[51] Fletcher R., Leyffer S. (2002): *Nonlinear programming without a penalty function*, Mathematical Programming, Vol. 91, 239–269

[52] Floudas C.A. (1995): *Nonlinear and Mixed-Integer Optimization*, Oxford University Press, New York, Oxford

[53] Floudas C.A., Pardalos P.M., Adjiman C.S., Esposito W.R., Gumus Z.H., Harding S.T., Klepeis J.L., Meyer C.A., Schweiger C.A. (1999): *Handbook of Test Problems in Local and Global Optimization*, Kluwer Academic Publishers

[54] Forrest J., de la Nuez D., Lougee-Heimer R. (2002): *Clp: An Open Source code for solving linear programming problems*, http://www.coin-or.org/Clp/userguide/index.html

[55] Fügenschuh A., Hiller B., Humpola J., Koch T., Lehmann T., Schwarz R., Schweiger J., Szabo J. (2011): *Gas network topology optimization for upcoming market requirements*, ZIB-Report 11-09, Zuse Institute Berlin, http://vs24.kobv.de/opus4-zib/frontdoor/index/index/docId/1234, to appear in Proc. of the 8th International Conference on the European Energy Market (EEM)

[56] Fukushima M. (1986): *A successive quadratic programming algorithm with global and superlinear convergence properties*, Mathematical Programming, Vol. 35, 253–264

[57] Geoffrion A. M. (1972) *Generalized benders decomposition*, Journal of optimization theory and applications, Vol. 10, No. 4, pp 237-260

[58] Goldfarb D., Idnani A. (1983): *A numerically stable method for solving strictly convex quadratic programs*, Mathematical Programming, Vol. 27, 1-33

[59] Gomory, R.E. (1958): *An algorithm for integer solutions to linear programs*, Princeton I.B.M. Mathematics Research Project, Technical Report No. 1, Princeton University, USA

[60] Gould N.I.M, Toint Ph.L. (2006): *Global convergence of a non-monotone trustregion filter algorithm for nonlinear programming*, in: Hager W.W., Huang S., Pardalos P.M., Prokopyev O.A. eds., Multiscale Optimization Methods and Applications, Springer, New York, pp. 125–150

[61] Grossmann I.E., Kravanja Z. (1997): *Mixed-integer nonlinear programming: A survey of algorithms and applications*, in: Conn A.R., Biegler L.T., Coleman T.F., Santosa F.N. (eds.): *Large-Scale Optimization with Applications, Part II: Optimal Design and Control*, Springer, New York, Berlin

[62] Grossmann I.E. (2002): *Review of nonlinear mixed-integer and disjunctive programming techniques*, Optimization and Engineering, Vol. 3, 227-252

[63] Grossmann I.E., Lee S. (2002): *Generalized convex disjunctive programming: nonlinear convex hull relaxation*, Computational Optimization and Applications, Volume 26, Number 1, 83-100

[64] Gupta O.K., Ravindran A. (1985): *Branch and bound experiments in convex nonlinear integer programming*, Management Science, Vol. 31, 1533-1546

[65] Hock W., Schittkowski K. (1981): *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer

[66] Jarre F., Stoer J. (2003) *Optimierung*, Springer

[67] Kelley J.E. (1960): *The cutting-plane method for solving convex programs*, Journal of the Society for Industrial and Applied Mathematics, Vol. 8, No. 4, pp. 703-712

[68] Kilinc M., Linderoth J., Luedtke J. (2010): *Effective separation of disjunctive cuts for convex mixed integer nonlinear programs*, Technical Report, Computer Sciences Department, University of Wisconsin-Madison

[69] Krumke O. (2006): *Integer programming*, Report, Technische Universität Kaiserslautern

[70] Lehmann T. (2006): *Quadratisch* L_1*-Optimierung mit Anwendungen auf Support-Vektor-Maschinen*, Diploma Thesis, University of Bayreuth, Germany

[71] Lehmann T., Schittkowski K. (2009): *MIQL : A Fortran subroutine for convex mixed-integer quadratic programming - user's guide, version 1.0*, Report, Department of Computer Science, University of Bayreuth

[72] Lehmann T., Schittkowski K. (2009): *MINLPB4 : A Fortran code for nonlinear mixed-integer quadratic programming by branch-and-bound - user's guide, version 1.0*, Report, Department of Computer Science, University of Bayreuth

[73] Lehmann T., Schittkowski K. (2009): *MISQPOA: A Fortran subroutine for mixed-integer nonlinear optimization by outer approximation - user's guide, version 1.0*, Report, Department of Computer Science, University of Bayreuth

[74] Lehmann T., Schittkowski K., Spickenreuther T. (2009): *BFOUR: A Fortran subroutine for integer optimization by branch-and-bound - user's guide* -, Report, Department of Computer Science, University of Bayreuth, Germany

[75] Leyffer S., Fletcher R. (1998): *Numerical experiments with lower bounds for MIQP branch-and-bound*, SIAM Journal on Optimization, Vol. 8, 604-616

[76] Leyffer S. (2001): *Integrating SQP and branch-and-bound for mixed-integer nonlinear programming*, Computational Optimization and Applications, Vol. 18, 295–309

[77] Li H.-L., Chou C.-T. (1994): *A global approach for nonlinear mixed discrete programming in design optimization*, Engineering Optimization, Vol. 22, 109–122

[78] Maratos N. (1978): *Exact penalty function algorithms for finite dimensional and control optimization problems*, Ph.D. Thesis, Imperial College Sci. Tech., University of London

[79] Mayne D.Q., Polak E. (1982): *A superlinearly convergent algorithm for constrained optimization problems*, Mathematical Programming Study, Vol. 16, 45–61

[80] McCormick G.P. (1976): *Computability of global solutions to factorable nonconvex programs: part I – convex underestimating problems*, Mathematical Programming, Vol. 10, pp.147-175

[81] Möller M. (2004): *Mixed integer models for the optimization of gas networks in the stationary case* Doctoral Thesis, Technische Universität Darmstadt, Germany

[82] Nowak I., Alperin H., Vigerske, S. (2005): *LaGO - An object oriented library for solving MINLPs*, International Series of Numerical Mathematics, Vol. 152

[83] Nowak I. (2005): *Relaxation and decomposition methods for mixed integer nonlinear programming*, International Series of Numerical Mathematics, Vol. 152, Birkhäuser,ISBN 978-3-7643-7238-5

[84] Omojokun E.O. (1989): *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints*, Ph.D. Thesis, University of Colorado at Boulder, USA

[85] Pan P. (2008), *A primal deficient-basis simplex algorithm for linear programming*, Applied mathematics and computation, Vol. 196, No. 2, 898-912

[86] Perregaard M. (2003): *Generating disjunctive cuts for mixed-integer programs*, Doctoral Dissertation, Carnegie Mellon University Pittsburgh, USA

[87] Powell M.J.D. (1983): *ZQPCVX, A Fortran subroutine for convex quadratic programming*, Report DAMTP/1983/NA17, University of Cambridge, England

[88] Powell M.J.D. and Yuan Y. (1986): *A recursive quadratic programming algorithm that uses differentiable exact penalty function*, Mathematical Programming, Vol. 35, 265–278

[89] Powell M.J.D. and Yuan Y. (1991): *A trust region algorithm for equality constrained optimization*, Mathematical Programming, Vol. 49, 189–211

[90] Quesada L., Grossmann I.E. (1992): *An LP/NLP based branch and bound algorithm for convex MINLP optimization problems* Computers & Chemical Engineering

[91] Ray T., Sarker R. (2007): *Genetic algorithm for solving a gas lift optimization problem*, Journal of Petroleum Science and Engineering, Vol. 59, 84-96

[92] Sahinidis N.V. (1996): *BARON: a general purpose global optimization software package*, Journal of Global Optimization, Vol. 8(2), pp. 201-205

[93] Schittkowski K. (1983): *On the convergence of a sequential quadratic programming method with an augmented Lagrangian search direction*, Optimization, Vol. 14, 197- 216

[94] Schittkowski K. (2003): *QL : A Fortran code for convex quadratic programming - User's guide*, Report, Department of Mathematics, University of Bayreuth, Germany

[95] Schittkowski K. (2010): *A collection of 100 test problems for nonlinear mixed-integer programming in Fortran - user's guide*, Report, Department of Computer Science, University of Bayreuth

[96] Schittkowski K., Yuan Y. X. (2010) *Sequential Quadratic Programming Methods* to appear: Wiley Encyclopedia of Operations Research and Management Science

[97] Selot A., Kuok L.K., Robinson M., Mason T.L., Barton P. (2008): *A short term operational planning model for upstream natural gas production systems*, AIChE Journal, 54, 495-515

[98] Sun X., Ruan N., Li D. (2006): *An efficient algorithm for nonlienar integer programming problems arising in series-parallel reliability systems*, Optimization Methods and Software, Vol. 21 617-634

[99] Tawarmalani M., Sahinidis N.V. (2005): *A polyhedral branch-and-cut approach to global optimization*, Mathematical Programming, Vol. 103, 225-249

[100] Thekale A., Gradl T., Klamroth K., Rüde U. (2009): *Optimizing the number of multigrid cycles in the full multigrid algorithm*, Report 09-5, Friedrich-Alexander-University Erlangen-Nürnberg, Institute for Computer Science

[101] Ulbrich S. (2004): *On the superlinear local convergence of a filter-SQP method*, Mathematical Programming, Vol. 100, 217–245

[102] Ulbrich M., Ulbrich S. (2003): *Non-monotone trust region methods for nonlinear equality constrained optimization without a penalty function*, Mathematical Programming Ser B, Vol. 95, 103–135

[103] Vardi A. (1985): *A trust region algorithm for equality constrained minimization: convergence properties and implementation*, SIAM Journal on Numerical Analysis, Vol. 22, 575–591

[104] Viswanathan J.,Grossmann I.E. (1990): *A combined penalty function and outer-approximation method for MINLP optimization*, Comput. chem. Engng., Vol. 14

[105] A. Wächter, L. T. Biegler (2006): *On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming*, Mathematical Programming 106(1), pp. 25-57

[106] Wang P., Litvak M. (2004): *Gas lift optimization for long-term reservoir simulations*, SPE Reservoir Evaluation & Engineering, Vol. 11, No. 1, 147-153

[107] Werner. J. (1999): Lecture notes on optimization, Institut für Numerische und Angewandte Mathematik, Universität Göttingen.

[108] Westerlund T., Petterson F. (1995): *An extended cutting plane method for solving convex MINLP problems*, Computers and Chemical Engineering, Vol. 21, 131-136

[109] Westerlund, Pörn (2002): *Solving pseudo-convex mixed-integer optimization problems by cutting plane techniques*, Optimization and Engineering, Vol. 3, 253-280

[110] Wojtaszek D. T. (2008): *Faster MIP solutions via new node selection rules* Ph.D. Thesis, Carleton university, NR40542

[111] Wolters K. (2006): *Implementation of cutting plane separators for mixed-integer programs*, Diploma Thesis, Zuse Institute Berlin, Germany

[112] Yuan Y. X. (1995): *On the convergence of a new trust region algorithm*, Numerische Mathematik, Vol. 70, No. 4, pp. 515-539

Acknowledgment

Finally, I would like to thank all people, that supported and accompanied me during the years of working at this thesis.

I am very thankful to my supervisor Prof. Dr. Klaus Schittkowski, who guided me in the right direction with regard to successful and interesting research topics. Furthermore, he supported me with his expertise, his patience and by sharing his international contacts in the optimization society. His high ambitions helped a lot to improve the scientific quality of this thesis. I would like to thank him for giving me the opportunity to work in the very interesting industrial cooperation with Shell on practical engineering problems.

I am also very thankful to my second supervisor Prof. Dr. Jörg Rambau, who always took the time for discussing my research results and who handed out good advises whenever possible.

The major part of my research was founded by the company Shell SIEP Rijswijk within the GameChanger Project IDC-2005050006. I am grateful for this support and I appreciated the work in a professional industrial environment on real world problems.

A special thanks goes to Prof. Dr. Christof Büskens at the University of Bremen for providing the possibility to work in his team on the development of a new nonlinear programming solver. I enjoyed the time in Bremen very much especially due to your good company Patrik, Matthias and Jörg.

I also want to thank my former colleagues from the Konrad-Zuse-Institute in Berlin. I had a very interesting time, in which I could improve my knowledge on optimization as well as my programming skills a lot. I am thankful for the new contacts in the optimization community and the work for EON on important future topics in the energy industry. Thank you Jesco, Jonas, Robert, Stefan, Benjamin, Armin, Thorsten and all the other guys from the ZIB for the nice time there.

Thanks to all members of the former team of Prof. Dr. Schittkowski for the nice time and friendly atmosphere. Thank you Sonja, Oliver, Axel, Björn, Thorsten and Mrs. Lachmann. Especially the scientific discussions with Oliver on MINLP solvers and related topics were very helpful.

Ein großer Dank gilt besonders meinen Freunden, die mich in den vergangenen Jahren stets unterstützt und aufgemuntert haben. Ihr hattet immer Verständnis dafür hatten, wenn ich ein Treffen nicht wahrnehmen konnte oder eine Party ausfallen lassen musste (es waren zum Glück nicht zu viele). Dies gilt besonders für meine Schulfreunde Steve, Basit, Stephan und Kahli aber auch für meine Studienfreunde, die mich viele Jahre an der Universität Bayreuth begleitet haben. Danke Sonja, Stefan, Peter, Armin, Jörg, Sascha, Tobi, Matthias, Steffi, Frank, Nicole und all die anderen (ehemaligen)

Bayreuther.

Mein größter Dank gilt aber meiner Familie, wobei ich mich besonders bei meinen Eltern und Geschwistern bedanken möchte, da Ihr mir immer zur Seite gestanden habt und mich immer unterstützt.

Thomas

Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt habe.

Des Weiteren versichere ich, dass ich die Hilfe von gewerblichen Promotionsberatern bzw. -vermittlern oder ähnlichen Dienstleistern weder bisher in Anspruch genommen habe noch künftig in Anspruch nehmen werde.

Ich versichere außerdem, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde.

Erlangen, den 15.04.2013

Thomas Lehmann
Anton-Bruckner-Str. 36
91052 Erlangen

Lebenslauf

Persönliche Daten

Thomas Lehmann

Geburtstag: 09.07.1980

Ausbildung

seit 02/2007	**Promotion** an der Universität Bayreuth On Efficient Solution Methods for Mixed-Integer Nonlinear and Mixed-Integer Quadratic Optimization Problems
10/2000 - 04/2006	**Studium der Wirtschaftsmathematik** an der Universität Bayreuth Abschluss: Diplom, Note: 1,2 Diplomarbeit: L1-Optimierung für Support-Vector-Machines
09/2003 - 07/2004	**Auslandsaufenthalt** Mathematikstudium an der Universität Exeter, England
07/1999 - 04/2000	**Wehrdienst** Flugabwehrraketenkanonier der Luftwaffe in Pfullendorf/Manching
09/1990 - 06/1999	**Schulische Ausbildung** am Arnold Gymnasium in Neustadt/Co Abschluss: Abitur, Note: 1,3
01/1999	Teilnahme am Wettbewerb **Jugend forscht** im Gebiet Arbeitswelt

Berufserfahrung

seit 02/2012	**Engineer** bei der Siemens AG
01/2010 - 12/2011	**Wissenschaftlicher Mitarbeiter** am Konrad-Zuse-Zentrums Berlin
02/2007 - 12/2009	**Wissenschaftlicher Mitarbeiter** der Universität Bayreuth
08/2006 - 12/2006	**Wissenschaftlicher Mitarbeiter** der Universität Bremen

Erlangen, 15. April 2013