EXPLOITING COMBINATORIAL RELAXATIONS TO SOLVE A ROUTING & SCHEDULING PROBLEM IN CAR BODY MANUFACTURING

JÖRG RAMBAU AND CORNELIUS SCHWARZ

ABSTRACT. Motivated by the laser sharing problem (LSP) in car body manufacturing, we define the new general routing and scheduling problem (RSP). In the RSP, multiple servers have to visit and process jobs; renewable resources are shared among them. The goal is to find a makespan-minimal scheduled dispatch. We present complexity results as well as a branch-and-bound algorithm for the RSP. This is the first algorithm that is able to solve the LSP for industrially relevant problem scales.

1. INTRODUCTION

The investigations in this paper were directly motivated by a real-world problem in car body manufacturing [12]: find a way to dispatch a set of laser welding robots so that the expensive laser sources can be used for more than one robot without exceeding the fixed total processing time (details will follow). Since any laser source can serve only one robot at a time, this poses a job-scheduling-dependent resource constraint on an otherwise routing based optimization problem, and the interesting objective is not the distance traveled or the weighted sum of delays but the makespan.

And this makes the problem mathematically extremely interesting. We do not know of any application context where an exact algorithm was proposed for a problem of this type. In this paper, we propose an algorithm for the general routing & scheduling problem (RSP); our application is a special case of it. The key is a carefully balanced mixture of branch-and-bound and polyhedral methods.

More specifically, our main result is an exact algorithm that solves the so-called *laser sharing problem* (formally defined below) to proven optimality or provably close to it for industry-relevant problem scales (≈ 30 welding jobs, ≤ 4 welding robots, ≤ 4 laser sources).

Now let us state in more detail (though still informally) what the laser sharing problem is about. Some car manufacturers use laser welding technology for the assembly of car bodies. The advantage is the possibility to weld along lines not only at single spots. We do not want to discuss the engineering pros and cons of this technique. Important for us is the following: In a welding cell there are two to six, usually four, industry robots simultaneously working on the around 30 welding jobs in a fixed process cycle time of around 30 s, given by the construction department. Traditionally, these robots are heuristically dispatched minimizing empty moves. If each robot has its own energy source for welding, a laser source, then it is enough to specify the assignment and order of jobs for each robot. However, in practice this leads to fairly large idle-times of the laser sources, namely during all the non-welding moves of the robots (the welding jobs are far from being connected).

A straight-forward idea is to let one laser source supply more than one robot with energy, but only one at a time. Since the total time spent in non-welding moves is fairly large, this is possible if the robots weld alternatingly. But this means: their movements are coupled by time-dependent resource requirements. A dispatch minimizing empty moves may be either infeasible in the presence of laser source sharing or it may exceed the processing time when waiting is used to respect all resource constraints. The resource constraints must be taken into account when assigning jobs to robots and planning the tours. Moreover, the orders of jobs in tours in not sufficient anymore to encode a solution, since it must be explicitely decided which robot welds when.

The overall goal is: find the minimal number of laser sources so that there exists a feasible dispatch, i.e., an assignment of jobs to robots, an assignment of robots to laser sources, tours for all robots through their assigned jobs, and a scheduling of all departure times satisfying the laser resource constraint so that the makespan does not exceed the given total processing time. Of course, this task can be accomplished by finding a makespan-minimal dispatch for any number of laser sources. And this, finally, is the *laser sharing problem* (collision avoidance is ignored for the moment).

Let us elaborate on three specific features of the laser sharing problem that – with or without collision avoidance – distinguish it from many usually studied problems:

(1) Robots must be *routed and explicitely scheduled* under shared resources;
(2) the *makespan* must be minimized in a multi-server system;
(3) to answer the question after the minimal number of necessary laser sources given the processing time, the *optimal makespan* must be computed and proved in the worst case.

First: The combination of the terms *routing* and *scheduling* appears in the literature very often. As early as 1965, there is a paper by S. Lin titled *The Vehicle Routing and Scheduling Problem* (unfortunately not available to us). However, as becomes clear in the following abundant literature (e.g., the often cited [21] and the survey [7]), we are faced with scheduling restrictions that depend only on restrictions at the customers (jobs in our case) by time windows. This may explain why the notion *scheduling* vanished again, and the vehicle routing and scheduling problem with time windows ended up as the vehicle routing problem with time windows.

In the terminology of Hempsch and Irnich [13], the time window constraints are all *intra-tour resource constraints*: whether or not a tour of a vehicle (a robot in our case) is feasible depends only on the scheduling of this tour alone. Moreover, often a mere ordering of customers in a tour already determines a unique cheapest scheduled tour (mostly by departuring as early as possible), so that an explicit scheduling (i.e., assigning departure times to the depot and all customers) can be avoided.

In our case, resource constraints are imposed by a resource shared among robots: the laser source. The feasibility of an individual tour of a robot is not well-defined anymore: we are facing an *inter-tour resource constraint*. Although in [13], the authors present a fairly general framework for inter-tour resource constraints such as global capacity restrictions and the like, their *giant tour method* will not straight-fowardly cope with global resources that can not be used more than once *at a time* – the laser source is a *renewable resource*. But most importantly, the giant tour representation seems to fail to encompass makespan optimization problems; at least, exact models based on this concept (let alone exact solution methods) did not come to our minds.

This brings us to the second point: *makespan minimization* – a common objective in scheduling theory – seems to be rarely studied in the integer linear programming (ILP) community, where the most successful exact algorithms for routing come from. It seems, there is an agreement that resource-constrained scheduling problems with makespan objectives are better handled in the scheduling world by special algorithms or direct branch-and-bound [6], whereas routing problems with linear objectives can be tackled very well by large-scale integer linear programming, mainly by column generation techniques along the lines of [7]. This works well even in real-time situations (see [14]), and apparently also for robot routing (see [22, 23]).

On the one hand, our preliminary tests on column generation methods for the laser sharing problem with makespan minimization were discouraging at best. There is a mathematical reason for this: although the makespan objective has a linear formulation in the LP-world (minimize an auxiliary variable bounded from below by all the individual tour lengths), the *integrality gaps* of ILPs based on this linear representation of the makespan are large in our experience, even for problems that are otherwise integral. And since the makespan objective depends on *all* tours in a solution, in any column generation approach we will encounter this large integrality gap in the Master Problem, which will give us – besides a lot of branch-and-bound nodes in the Master ILP – misleading dual values for the pricing problem. We will elaborate on the quality of bounds in Section 5. Of course, one could restrict the tour length in a column generation approach to the given processing time and not care about the optimization objective. But column generation methods usually require reasonably good feasible solutions right from the start; we have yet to see a column-generation algorithm that can prove infeasibility fast.

On the other hand, lower bounds in branch-and-bound algorithms in resource constrained scheduling are often generated by precendence constraints on the processing orders of jobs (project scheduling, see [5]). In our case, there is *no other scheduling constraint* other than that the resources can not be used more than once at a time. Reasonable lower bounds that are good enough to prove optimality without full enumeration must incorporate routing delays – mainly absent in results on makespan minimization in (project) scheduling.

And this touches already the third point: *proven optimality* is required in order to conclusively decide whether or not we can finish with a given number of laser sources within the processing time. In many applications like airline crew scheduling this is neither possible nor desirable: the data is not exact anyway. In our case, the data about the times the laser source is used is quite reliable, and optimality is at least in principal substantial for the minimization of the number of laser sources necessary. Thus, we want to achieve proven optimality in as many cases as possible, and otherwise optimality gaps that are as small as possible. Since the laser sharing problem is a strategic planning task, computation times of a day or so are acceptable.

In this paper we will show how a combination of the strengths of all mentioned approaches can indeed provide us with an algorithm that can solve industrial-scale instances of the laser sharing problem. We use an outer branch-and-bound based on bounds coming from the solutions of NP-hard subproblems, solved by means of ILP techniques. The investigation of the relations between various subproblems of the laser sharing problem plays a key role in the design of bounding computations.

Why do we deliberately choose to solve NP-hard subproblems? Branch-and-bound is kind of a divide-and-conquer approach; thus, one has to balance the difficulty of the subproblems with the difficulty of the tree exploration. Only if the subproblems capture at least one part of the problem structure faithfully, the remaining parts of the problem can be hunted down in the enumeration. The LP-relaxations of global models that we investigated either leave too much work for the enumeration or are not polynomial either.

Why do we succeed in solving these subproblems fast enough? The important observation is that a scale that is large for our main problem will pose instances of comparably small scale for the NP-hard subproblems – given the state-of-the art ILP techniques.

Our algorithm is able to cope with another side constraint without which the solution method would not be convincing: *collision avoidance*. We tried hard to find a generally accepted input data concept for collisions between industry robots, and we could not find anything but simulation by means of commercial tools. This, however, can not be used as a model in an exact solver. Thus, we suggest a collision model based on a classification of collisions into line-line-collisions and line-point-collisions that have to be avoided in the most conservative fashion. Meanwhile, this model has partly been adopted in [22, 23].

Short, preliminary versions of the algorithm for the laser sharing problem without collision avoidance have already been presented together with preliminary computational results in [17]. The incorporation of collision avoidance required the extension of the Tuchscherer model from [12]. The resulting algorithm is yet cleaner and more powerful than the older version. A preliminary version with collision avoidance together with preliminary computational results was presented in [20] and formally introduced in [18].

In this paper, our comprehensive study of the laser sharing problem goes beyond the conference presentations in [17, 18] in the following aspects: We present

- how the laser sharing problem with collision avoidance fits into a new abstract framework for resource constrained routing and scheduling problems (RSP) with the objective to minimize the makespan;
- new complexity results for the RSP and natural subproblems; in particular, a pseudo-polynomial algorithm for the solution of RSP-T, i.e., the RSP with fixed tours, where the number of servers and resources is constant but the number of jobs is variable;
- a generic algorithm for the general RSP, based on the exact solution of NP-hard subproblems inside a combinatorial branch-and-bound scheme;
- comprehensive computational results for our algorithm applied to the laser sharing problem with laser and space resources; moreover, comparisons to the direct MILP-solution of alternative models that follow various common modelling principles; the data is based on an industry-standard simulation tool and a geometrically plausible (yet artificial) set of welding jobs.

The practically most relevant result can be summarized as follows: our RSP-algorithm is the first one that can solve a large part of our industrial-scale benchmark instances of the laser sharing problem to proven optimality. Bounds obtained by the exact solution of the NP-hard subproblems (combinatorial relaxations) are significantly tighter and at the same time much faster to compute than bounds based on LP-relaxations of reasonable MILP models.

A disclaimer is in place here: although our algorithm is formulated in the abstract setting of the RSP, we can not expect such a good performance for the general RSP. Our algorithm exploits the specific properties of relevant laser sharing instances. Its mechanism heavily relies on the fact that in the laser sharing problem the laser sources are never the bottleneck – otherwise there would be no incentive to reduce the number of laser sources in the first place. We believe that taylor-made RSP algorithms tuned for other special applications can take profit of a careful bottleneck analysis as well; this seems to be the key to find the combinatorial relaxations that provide the best bounds.

The paper is organized as follows: In Section 2 we formally introduce the laser sharing problem and our new concept for collision avoidance. Moreover, our abstract framework for resource constrained routing and scheduling problems RSP is presented. Section 3 is devoted to complexity results like the new pseudo-polynomial algorithm for the RSP-T. Next, in Section 4 we describe the ideas, the building blocks, and the overall design of the new algorithm CBB. In Section 5 we present computational results based on data obtained by a professional robot simulation tool by KuKa. We compare the quality of bounds generated by the new algorithm with seemingly more straight-forward approaches. Finally, we draw conclusions in Section 6.

2. Problem Statement

We start with a formulation of the laser sharing problem (LSP) before we give the definition of the more general routing & scheduling problem (RSP).

2.1. **The Laser Sharing Problem.**

An instance of the *Laser Sharing Problem* LSP consists of a set S of *robots* (the *servers*), a set J of *jobs*, a set $J^s \subseteq J$ of feasible jobs for every $s \in S$, a set L of *laser sources*, a set C_{ll} of *line-line collisions*, a set C_{lp} of *line-point collisions*, and a distance table δ.

Each robot $s \in S$ has a *depot position* d^s where the tour has to start and to end. Each job $j \in J$ has two *job positions* j_a, j_b. *Processing* a job means to traverse (j_a, j_b) or (j_b, j_a). For simplicity we define $Q := \{d^s \mid s \in S\} \cup \{j_a, j_b \mid j \in J\}$ to be the set of all positions.

The shortest time Server s needs to move from Position p to Position q is referred to by $\delta^s(p,q) \geq 0$. If p and q are end positions of the same welding job, then this move is done in welding mode processing the corresponding job; otherwise it is done in driving mode. Our distances are assumed to satisfy the triangle inequality. When Laser Source l switches robots, then there is a delay of $\tau^l \geq 0$.

An entry $(s_1, p_1, q_1, s_2, p_2, q_2) \in C_{\text{ll}} \subseteq S \times Q \times Q \times S \times Q \times Q$ is interpreted as follows. There exists two start times t_1, t_2 with: if Robot s_i starts a move from p_i to q_i at t_i, then a collision between s_1 and s_2 will happen. Our conservative restriction is that in this case the two moves (s_1, p_1, q_1) and (s_2, p_2, q_2) must not overlap in time.

Similarly, $(s_1, p_1, q_1, s_2, p_2) \in C_{\text{lp}} \subseteq S \times Q \times Q \times S \times Q$ means Robot s_2 residing at Position p_2 will collide with Robot s_1 moving from p_1 to q_1. Here, we do not allow Robot s_2 to visit Position p_2 while s_1 is moving from Position p_1 to Position q_1.

In the following, bars on symbols indicate decisions. The task is to assign to each robot $s \in S$ a set of jobs $\bar{J}^s$ to process, a tour $\bar{T}^s = (d^s = q_1^s, \ldots, q_{n^s}^s = d^s)$ through the corresponding job positions, a laser source $\bar{r}^s \in L$ used for welding, and a schedule $\bar{t}^s$ that assigns a departure time $\bar{t}^s(q)$ to each visited position q in the tour $\bar{T}^s$ such that

- each robot s is assigned only feasible jobs, i.e., $\bar{J}^s \subseteq J^s$;
- each job j is processed in exactly one tour $\bar{T}^s$;
- jobs assigned to robots s_1, s_2 sharing a laser source $\bar{r}^{s_1} = \bar{r}^{s_2}$ do not overlap in time with respect to the corresponding schedulings $\bar{t}^{s_1}$ and $\bar{t}^{s_2}$;
- all robot moves are collision free in the sense described above with respect to the corresponding schedulings.

The completion time of each robot is the time it arrives back at d^s according to the schedule of its tour. The goal is to minimize the makespan, which is the maximum over the completion times of the robots.

We do not formalize this any further, since the LSP is a special case of the general routing & scheduling problem (RSP) in the next subsection.

Remark 1. We do not allow preemption. This means: whenever a robot starts driving or welding from one position to another, then the whole move is done at once. Therefore, waiting is only possible at the depot position or at one of the job positions.

Remark 2. Any line-point collision $(s_1, p_1, q_1, s_2, p_2) \in C_{\text{lp}}$ induces for every $q_2 \in Q$ the line-line collisions $(s_1, p_1, q_1, s_2, p_2, q_2) \in C_{\text{ll}}$ (set $t_2 = 0$ in the definition) and $(s_1, p_1, q_1, s_2, q_2, p_2) \in C_{\text{ll}}$ (revealed by setting $t_2 := \delta^s(q_2, p_2)$). Thus, line-point collisions always induce line-line collisions.

As special cases we introduce

- LSP-J where we assume that an assignment $\bar{J} : S \to 2^J$ of jobs to robots is fixed.
- LSP-T where in addition to LSP-J also a tour $\bar{T}^s$ for every robot s is given and thus only an optimal schedule has to be found.
- LSP-s where only a single robot is present, so resource-sharing is void.

2.2. **The Routing & Scheduling Problem.** In this section we propose a generic model called *Routing & Scheduling Problem (*RSP*)* that encompasses the LSP as well as some other very general routing/scheduling problems from the literature: for example, the windy rural postman problem [2, 3], project scheduling with renewable resources $(Sm, \sigma, 1 | \cdot | C_{\max})$ – see [4] for the notation –, and the machine scheduling problem $(Rm \mid M_j,\, s_{i,j,k} \mid C_{\max})$ – see [16] for notation.

Particularly interesting is the fact that in the LSP collision avoidance can be modeled by resource sharing very similar to laser source sharing. We will again indicate all structures modelling decisions with a bar on top of the symbol: the assignment of jobs to servers $\bar{J}^s$ specifies for each server all jobs assigned to it, the tour $\bar{T}^s$ determines along which route Server s moves through its jobs; given the tour, the choice function $\bar{r}^s$ determines which resource of a certain type is selected by Server s for its tour; the scheduling function $\bar{t}^s$ determines at which time Server s leaves a position.

An instance of the RSP consists of the following data: Let S be the set of all *servers*, and let J be the set of all *jobs*. Denote by $J^s \subseteq J$ the set of *feasible jobs* for a server $s \in S$, i.e., the subset of jobs for which Server s has the necessary capabilities. Each server has a *depot position* d^s. Each job $j \in J$ is associated with two special *job positions* j_a, j_b. Its *possible request arcs* are the pairs (j_a, j_b) and (j_b, j_a). A server *s serves* one of its feasible jobs $j \in J^s$ if it traverses either Arc (j_a, j_b) or Arc (j_b, j_a). For each server s, let $G^s = (V^s, A^s)$ be its *graph of feasible moves*, i.e., the complete directed graph consisting on all positions of feasible jobs and the position d^s of the depot.

Let $\delta^s : A^s \to \mathbb{R}_+ \cup \{\infty\}$ be the (in general asymmetric) *distance function* of Server s, i.e., $\delta^s(v, w)$ is the time Server s needs to move from v to w. For $(v, w) = (j_a, j_b)$ or $(v, w) = (j_b, j_a)$ for some job j, the travel times are the corresponding *job processing times*. The job processing time may depend on the direction, i.e., $\delta^s(j_a, j_b) \neq \delta^s(j_b, j_a)$ is possible.

Let $\mathscr{R}$ be the set of all *resource types*, where each resource type $R \in \mathscr{R}$ is a set of one or more *resource type instances* $r \in R$, *resources*, for short. The type of Resource r is denoted by $R(r)$. Each resource can be used by only one server at a time. Before a resource r currently used by Server s_1 can be used by Server s_2, a *switching time* of $\tau(r, s_1, s_2)$ has to pass. We denote by $\hat{R} := \bigcup_{R \in \mathscr{R}} R$ to the set of all resources.

Let $\rho^s : A^s \cup V^s \to 2^{\mathscr{R}}$ be the *resource-demand function* of Server s, i.e., $\rho^s(v, w) \subseteq \mathscr{R}$ is the set of resource types required by Server s while traversing Arc (v, w), and $\rho^s(v) \subseteq \mathscr{R}$ corresponds to the set of resource types that Server s needs while it resides at Node v. We assume $\rho^s(v) \subseteq \rho^s(a)$ for every node v and every arc $a = (v, w)$ or $a = (w, v)$.[1] For a subgraph H^s of G^s let $\rho^s(H^s)$ be the set of resource types needed by the arcs and nodes in H^s.

[1]Resources are demanded for an arc if they are needed somewhere on the arc, in particular, if they are needed on one of the end nodes of the arcs.

The task is to generate the following output data for each server s: A *tour* $\bar{T}^s$ (through a subset of the jobs $\bar{J}^s$), together with a *resource selection* $\bar{r}^s$ and a *scheduling* $\bar{t}^s$ for it so that

- each job is processed in exactly one tour;
- no *resource conflict* occurs;
- the *makespan* is minimal.

This means, in more detail:

A *job assignment* specifies for each server $s \in S$ a set of jobs $\bar{J}^s$. This is the set of jobs to be processed by s. For a given job assignment $\bar{J}^s$ of jobs to Server s, a *routing* is given by sequencing the end positions of jobs in $\bar{J}^s$.

The result is a *tour* $\bar{T}^s = (d^s = q_1^s, \ldots, q_{n^s}^s = d^s)$ in V^s starting and ending at the depot of Server s connected by arcs in A^s. We occasionally consider such a tour as a subgraph $\big(V^s(\bar{T}^s), A^s(\bar{T}^s)\big)$ of G^s. When a tour $\bar{T}^s$ is given, the induced set of visited jobs is denoted by $\bar{J}^s$, if the tour is clear from the context, and otherwise as $J(\bar{T}^s)$.

A *dispatch* is a collection of tours $(\bar{T}^s)_{s\in S}$, one for each server. A dispatch $(\bar{T}^s)_{s\in S}$ *partitions a set of jobs J* if J is the disjoint union of all $J(\bar{T}^s)$ over all $s \in S$.

A *resource selection* for a tour $\bar{T}^s$ of Server s is a choice function $\bar{r}^s : \rho^s(\bar{T}^s) \to \hat{R}$, i.e., $\bar{r}^s(R) \in R$ for all $R \in \rho^s(\bar{T}^s)$. In words: $\bar{r}^s$ chooses a particular resource for each resource type needed by Server s in its tour.[2] Although the resource selection depends on the choice of a tour, we ignore this in the notation of "$\bar{r}^s$" to avoid unnecessary clutter.

A *scheduling* for a tour $\bar{T}^s = (d^s = q_1^s, \ldots, q_{n^s}^s = d^s)$ of Server s specifies for all $i = 1, \ldots, n^s$ non-negative departure times $\bar{t}_i^s$ with $\bar{t}_{i+1}^s \geq \bar{t}_i^s + \delta^s(q_i^s, q_{i+1}^s)$ for all $i = 1, \ldots, n^s - 1$, i.e., it chooses a routing-consistent departure time of the server in each position of its tour. It is convenient to consider the "departure" time from the last (depot) position as the completion time. Again, we leave out the dependence of the choice of a tour in the notation of "$\bar{t}^s$".

Resource selections $\{\bar{r}^s\}_{s\in S}$ and schedulings $\{\bar{t}^s\}_{s\in S}$ for a dispatch $\{\bar{T}^s\}_{s\in S}$ incur a *resource conflict* if there are two servers $s_1, s_2 \in S$ using the same resource at the same time, i.e., the following happens:

(1) There is an arc (p_i, p_{i+1}) of $\bar{T}_{s_1}$ with one of the following:
  (a) there is an $R \in \rho^{s_1}(p_i, p_{i+1}) \cap \rho^{s_2}(d^{s_2}) \neq \emptyset$, or
  (b) there is an arc (q_j, q_{j+1}) of $\bar{T}^{s_2}$ and an $R \in \rho^{s_1}(p_i, p_{i+1}) \cap \rho^{s_2}(q_{j+1}) \neq \emptyset$, or
  (c) there is an arc (q_j, q_{j+1}) of $\bar{T}^{s_2}$ and an $R \in \rho^{s_1}(p_i, p_{i+1}) \cap \rho^{s_2}(q_j, q_{j+1}) \neq \emptyset$.

(2) $\bar{r}^{s_1}(R) = \bar{r}^{s_2}(R)$;

(3) (a) $\big([0, \bar{t}_1^{s_1}] \cup [\bar{t}_{n^{s_1}}^{s_1}, \infty)\big) \cap [\bar{t}_j^{s_2}, \bar{t}_j^{s_2} + \delta^{s_2}(q_j, q_{j+1})] \neq \emptyset$, or
  (b) $[\bar{t}_j^{s_2} + \delta^{s_2}(q_j, q_{j+1}), \bar{t}_{j+1}^{s_2}] \cap [\bar{t}_i^{s_1}, \bar{t}_i^{s_1} + \delta^{s_1}(p_i, p_{i+1})] \neq \emptyset$, or
  (c) $[\bar{t}_i^{s_1}, \bar{t}_i^{s_1} + \delta^{s_1}(p_i, p_{i+1})] \cap [\bar{t}_j^{s_2}, \bar{t}_j^{s_2} + \delta^{s_2}(q_j, q_{j+1})] \neq \emptyset$, resp.

Resource selections and schedulings for a dispatch with no resource conflict are called *conflict-free*.

Summarized, a *feasible solution to the Routing & Scheduling Problem (*RSP*)* consists of a dispatch partitioning all jobs, together with conflict-free resource selections and schedulings. We call the tupel $\bar{D} = (\bar{T}^s, \bar{r}^s, \bar{t}^s)_{s\in S}$ a *scheduled dispatch*.

A scheduling $\bar{t}^s$ determines the *completion time* of a tour $\bar{T}^s$, which equals $\bar{t}_{n^s}^s$, i.e., the "departure time" at the final (depot) position of the tour, which was declared to be the completion time of the tour. The *makespan* of a dispatch $\{\bar{T}^s\}_{s\in S}$ with respect to schedulings

[2]Note that the resource assignment is not time-dependent. This could be changed at the cost of further notational complications. Since we will not use it, we stick to time-independent resource assignements.

$\{\bar{t}^s\}_{s\in S}$ equals $\max_{s\in S}\bar{t}^s_{n^s}$, i.e., the maximum over all completion times. The objective is to find a feasible solution with minimal makespan.

Remark 3. The reader may ask why not all node resources can be modeled in terms of arc resources by splitting nodes into two with an arc inbetween. Indeed, resources required at a (geometric) position for a certain task taking a specified amount of time (reload winter-gritting truck, maintain aircraft, ...) should be modeled as arc resources that way. The reason why we need node resources comes from the fact that waiting for resources is only possible at nodes, and this waiting option for an unspecified time may need resources as well, but for a time depending on the solution, i.e., that is not specified in advance (parking spot for trucks, space for robots ...). We rule out all other options to delay a tour, like a lower speed on arcs or detours. In models based on discrete times and a time-expanded network, hold-over arcs could model these waiting options at nodes. But we did not want to fix a model requiring discrete times in the problem statement.

Example 1 (LSP without collision avoidance). The LSP with collisions ignored can easily be modeled as an RSP: There is one resource type, namely "laser source", with as many resources as we have laser sources installed. The resource selection models that robots cannot switch among the laser sources. The rest is straight-forward.

Example 2 (LSP with Collision Avoidance). In the context of RSP, we can model collisions as additional resources: For every line-line collision $C_{s_1,p_1,q_1,s_2,p_2,q_2} \in C_{ll}$ we introduce a resource type $R = R^{s_1,p_1,q_1,s_2,p_2,q_2}$ consisting of one resource $r = r^{s_1,p_1,q_1,s_2,p_2,q_2}$ with $\tau(r,s_1,s_2) = 0$. Then we add r to $\rho^{s_i}(p_i,q_i)$. Line-point collisions are handled analogously. The resources can be viewed as parts of space that can only be used by a single (moving) server at a time.

Example 3 (The Windy Rural Postman Problem). Given an undirected graph $G = (V,E)$, two edge costs functions $c^1,c^2 : E \to \mathbb{R}\cup\{\infty\}$, a set $E_R \subseteq E$ of required edges, the windy rural postman problem (WRP) asks for a minimum cost closed walk in G using all required edges at least once. The cost of an edge e in the walk is either c^1_e or c^2_e depending on the direction e is traversed. We now show how to model WRP as a "routing-only" RSP: Set $S := \{s\}, \mathscr{R} := \emptyset$, interpret every edge $e = (v,w) \in E_R$ as a job j with the two job positions v and w. Set d^s to an arbitrary node incident with some edge $e \in E_R$. For all $v,w \in V$, we set $\delta^s(v,w)$ to the length of a shortest v-w-path in G. An optimal solution to the WRP can be derived from an optimal solution to the RSP by expanding all arcs in the server tour of the RSP solution to shortest paths again.

Example 4. Consider the following project scheduling problem $(PSm,\sigma,1|\cdot|C_{\max})$ given in the Graham notation [11] as extended to project scheduling in [4]. Here we have m renewable resources each with amount σ and every job needs at most one of them. This problem can be modeled as a "scheduling-only" RSP as follows: Assign to each job j two artificial positions j_a, j_b. Introduce a server s for every job j with $d^s := j_a$ and set $J^s = \{j\}$. Set $\delta^s(d^s, j_a) := 0, \delta^s(j_a, j_b) := p_j, \delta^s(j_b, d^s) := 0$ for this server, where p_j denotes the processing time of Job j. Set all other distances to ∞. Set $\mathscr{R} := \{R_1,\dots,R_m\}$ with $R_i := \{R_{i,1},\dots,R_{i,\sigma}\}$ and set ρ according to the resource requirements of the jobs. Then, the minimum makespan solution of RSP corresponds to a minimum makespan solution of $(PSm,\sigma,1|\cdot|C_{\max})$.

Remark 4. In our definition, a tour of a server visits only the end positions of the jobs assigned to it, and each of them exactly once. No position can be used twice, and no other position can be used for the routing of the server. When there are no resource conflicts, then visiting an additional position can not improve the makespan anyway. However, when a resource conflict on a move between jobs needs to be resolved, visiting additional positions ("moving around") can be useful: the additional time for a conflict-free detour may by smaller than the time required to wait for the availability of a resource. That is, the triangle inequality for the marginal increase of the makespan does not hold. The problem can be captured by adding Steiner positions to the problem. These are positions where a visit is optional. But introducing Steiner positions makes the problem much harder to solve. Therefore, we do not allow detours through additional positions.

The generalization of the special cases of LSP-J, LSP-T, LSP-s will be denoted by RSP-J (fixed partition of jobs among servers), RSP-T (fixed server tours), and RSP-s (single-server problem).

3. Complexity Results

In this section, we partly clarify the complexity of the LSP and the RSP, resp., and some corresponding subproblems.

Proposition 1. *The single-server laser-sharing problem LSP-s is NP-hard in the strong sense.*

Proof. We will reduce the asymmetric traveling salesman problem (ATSP) to LSP-s: Let $G = (V,A)$ be a directed graph with costs $c : A \to \mathbb{R}_+$. The ATSP asks for a minimal tour in G visiting all cities in V exactly once. In order to create an instance of LSP-s we set $J := V$ with artificial positions j_a, j_b and distances $\delta^s(j_a, j_b) := \delta^s(j_b, j_a) := 0$. For two distinct jobs $i, j \in J$ we set $\delta^s(i_e, j_s) := c_{i,j}$ for $s, e \in \{a,b\}$. An optimal solution for this LSP-s problem coincides with an optimal ATSP tour. Furthermore, this transformation is clearly pseudo polynomial. □ □

If for all jobs processing times $\delta^s(j_a, j_b) = \delta^s(j_b, j_a)$ holds then we can shrink j_a, j_b to one node obtaining a standard ATSP. If in addition the distances are of type $\delta^s(p,q) = |a_p - b_q|$ with two parameters a_v, b_v associated with each node v then LSP-s can be solved in $\mathscr{O}(n^2)$ (see [16] chapter 4.4 for the algorithm) where n is the number of ATSP nodes, i.e., $|J| + 1$. An example of such a situation is the case where Server s corresponds to some painting machine, and a_v, b_v are cleanup and refill time of the colors. However, in general this is not the case for industrial robots.

When $|L| = 1$ and $\tau^l = 0$ then the LSP-T problem reduces to a single machine scheduling problem with precedence constraints and minimal time lags. The precedence constraints form chains – one per robot – and the time lag between two jobs is due to the driving time of the robot. In the Grapham notation [11], this is called $(1 \mid \text{chains}(l_{i,j}) \mid C_{\max})$. Wikum et al. (see [25, Proposition 3.4]) proved this problem to be NP-hard in the strong sense, if each chain contains at least two jobs. Because our robots have to go home after their last job, we can sharpen this result to one welding job per robot:

Proposition 2. *The LSP-T is NP-hard in the strong sense, even when every robot welds only one job.*

Proof. We will reduce Sequencing With Release Times And Deadlines, which is NP-hard in the strong sense (see Garey and Johnson [10]), to the decision version of LSP-T. An instance of this problem consists of a set J' of jobs. Each job $j \in J'$ has an associated processing time p_j, a deadline d_j, and a release time r_j. The question is whether there exists a single machine schedule in which all jobs are scheduled in time, i.e., an assignment $j \mapsto x_j$

of jobs to start times x_j so that $x_j \geq r_j$, $x_j + p_j \leq d_j$ for all j, and all intervals $[x_j, x_j + p_j]$ are pairwise disjoint.

Given such an instance, we can construct an instance of the decision version of LSP-T as follows: An upper bound for the makespan objectiv is given by $\mu := \max_{j \in J'} d_j$. For each job $j \in J'$ we introduce a robot s with an artificial depot position d^s and a welding job j with two artificial end positions j_a, j_b. We set the distances $\delta^s(j_a, j_b) := p_j$, $\delta^s(d^s, j_a) := r_j$ and $\delta^s(j_b, d^s) := T - d_j$. The tour of robot s contains only Job j, i.e., $T_s := (d^s, j_a, j_b, d^s)$. Collisions are not needed, $C_{\text{ll}} := C_{\text{lp}} := \emptyset$. Because we model a single machine scheduling problem, we only introduce one laser source $L := \{l\}$ with $\tau^l := 0$.

Then, there exists an solution for LSP-T with makespan not greater than μ if and only if there exist a feasible schedule for SEQUENCING WITH RELEASE TIMES AND DEADLINES. Clearly, this transformation is pseudo-polynomial in time and input size. □ □

We remark that there is a polynomially solvable special case $(1 \mid \text{chains}(l), p_j = p \mid C_{\max})$ with constant driving time l and constant processing time p [15].

Because LSP-s and LSP-T are NP-hard in the strong sense, this is also true for LSP-J and LSP and the corresponding RSP versions, since all of them contain the first two as special cases.

Corollary 1. *The decision versions of LSP-J,* LSP, RSP*, RSP-s, RSP-T, and RSP-J are all NP-hard in the strong sense.* □

In the proof of Proposition 2 we used one robot for each job. But in practice the situation is exactly the opposite: few robots have to process many jobs. This raises the question whether the complexity changes if the number of robots is fixed and the number of jobs is not.

For the special case $(1 \mid 2 - n_1 n_2 \text{chains}(l_{i,j}) \mid C_{\max})$, Wikum [24] proposed a dynamic programming algorithm that runs in $\mathcal{O}(n_1 n_2^2 (1 + \max\{L_2 - p_1, 0\})$, where $L_2 = \max_j l_{2,j}$ and $p_1 = \min_i p_{1,i}$. Here, $l_{i,j}$ is the minimal delay between the j-th and the $j+1$-th job of Chain i, and $p_{i,j}$ denotes the processing time the j-th job in Chain i. Wikum used the following solution representation, on which he built his algorithm: Fix Chain 2, and for each consecutive pair of jobs $J_{2,j}, J_{2,j+1}$, decide how many jobs of Chain 1 are processed in between. The disadvantage of this representation is that it cannot be extended to handle more than two chains.

In the sequel, we get to one of our main results: we describe a pseudo-polynomial dynamic programming algorithm for $(1 \mid k - n_1, \ldots, n_k \text{chains}(l_{i,j}) \mid C_{\max})$ with any fixed number of chains. Moreover, the algorithm is stated for the more general RSP-T with any fixed number of resources: In constrast to the one machine scheduling problem, RSP-T faces multiple resources. If there is only a fixed number of resources, then all possible resource assignments can be enumerated. In the context of LSP-T this assumption reflects the situation where there are only few laser sources and few possible collisions compared to the number of jobs, which is typical in practice.

In the following, we first assume that the resource assignments $\bar{r}^s$ are fixed. The general algorithm then runs the fixed-resource-assignement algorithm for every possible resource assignment and chooses the best solution. As usual in scheduling, we will assume all data to be integer.

In order to provide a solid notational environment for a dynamic programming algorithm, we will now formulate RSP-T as a discrete-time dynamic control problem (see [8] for a textbook about dynamic control problems). Let $(\bar{T}^s)_{s \in S}$ be the given tour set with

$\bar{T}^s = (q_1^s, \dots, q_{n^s}^s)$. Here $q_1^s = q_{n^s}^s = d^s$ and q_i^s, $i = 2, \dots, n^s - 1$ are alternating job start and job end positions.

In order to construct a discrete-time dynamic control problem we have to specify a finite set I of stages. Moreover, we need to define for every stage $i \in I$ a state space X_i containing states x_i, a control space U_i containing controls u_i, a set of feasible controls $U_i(x_i) \subseteq U_i$ for every state x_i, a system dynamics $x_{i+1} = f_i(x_i, u_i)$, a stage cost function $g_i\big(x_i, u_i\big)$ for $i = 0, 1, \dots, N-1$, and a terminal cost $g_N(x_N)$.

The task in the control problem is to find an optimal control policy, i.e., a sequence of decision rules $\mu = (\mu_0, \mu_1, \dots, \mu_{N-1})$ with $\mu_i : X_i \to U_i$ such that for a given start state x_0 the cost $V_\mu(x_0) := g_N(x_N) + \sum_{i=0}^{N-1} g_i\big(x_i, \mu_i(x_i)\big)$ is minimal among all feasible policies.

In order to formulate the RSP-T as such a control problem, we use event-based stage indices as follows: the number i of already left positions is taken as the *stage index* of the dynamic control problem. Then, the *horizon* is $N := \sum_{s \in S} n^s$. Consequently, the *set of stages* is $I = \{0, \dots, N\}$. For each $i \in I$ the set of *feasible states* is defined as

$$X_i := \{(i_s, w_s, o_r, d_r, e_r)_{\substack{s \in S \\ r \in \hat{R}}} \in X \mid i = \sum_{s \in S} i_s\},$$

where the state components $(i_s, w_s, o_r, d_r, e_r)$ are defined in the sequel.

The *index* state component $i_s \in \{0, \dots, n^s\}$ is the number of positions that Server s has already left in its tour $(d^s = q_1^s, \dots, q_{n^s}^s = d^s)$. Here "$i_s = 0$" means that the server is waiting for the availability of resources for its move from the depot to the first job position. The value "$i_s = n^s$" means that the server has already arrived back at the depot. Any other value of i_s means that the server has left position $q_{i_s}^s$. We will write δ^s for $\delta^s(q_{i_s}^s, q_{i_s+1}^s)$ for $i_s = 0, \dots, n^s - 1$ and $\delta^s := 0$ for $i_s = n^s$.

The idea is to encode a feasible solution of RSP-T in the control problem as an order in which the servers gain access to the resources they need. The control decision which server is next in this order takes place whenever some server reaches the end position of its current move. Be careful: this does not mean that Server u gains access to the resources immediately but only after it is in the right position and all resources are available. Therefore, in order to keep track of the marginal influence of this decision on the makespan we need additional state components for time measurements. In order to find out when Server u can continue to work, we need to know when the required resources are free.

To this end, for each resource r the *delay* state component $d_r \in \{0, \dots, M\}$ indicates when the Resource r will be available again. M can always be choosen as an upper bound for the optimal makespan, e.g., $M := \sum_{s \in S} \sum_{i=1}^{n^s - 1} \delta^s(q_i^s, q_{i+1}^s)$.

When d_r is zero, we still have to keep track of the switching time: for this purpose, the *enabling* state component $e_r \in \{0, \dots, \max_{s_1, s_2 \in S, s_1 \neq s_2} \tau(r, s_1, s_2)\}$ denotes for how long Resource r has been idle since its last use.

Futhermore, the *owner* state component $o_r \in S \cup \{\sigma\}$ denotes the last server that was granted access to r (specified by a corresponding previous control). Here, the special symbol σ denotes "nobody" for the case that a resource has not been reserved so far at all. We set $\tau(r, \sigma, s) := 0$ for all $s \in S$.

Since a control decision will take place whenever some server reaches the end position of its current move, we store in the *will-arrive* state component $w_s \in \{0, \dots, M\}$ the time Server s needs to reach its next position. This duration includes possible waiting times underway for resources becoming available.

Our control descision $\mu_i(x_i)$ selects the next server getting exclusive access to all needed resources. We therefore set the set of possible controls to $U_i := U := S$.

When server s is selected to move from q_{i_s} to q_{i_s+1}, then any resource $r \in \rho^s(q_{i_s+1})$, e.g., a resource of a line-point collision, blocks all other servers requiring r afterwards, until s is choosen by μ again. Note that r is blocked already before s arrives at q_{i_s+1} because of our general assumption $\rho^s(q_{i_s+1}) \subseteq \rho^s(q_{i_s}, q_{i_s+1})$. Thus, we demand that $\mu_i(x_i) \in U_i(x_i)$ with

$$U_i(x_i) := \{u \in U \mid i_u < n^u - 1 \wedge \rho^u(q^u_{i_u+1}, q^u_{i_u+2}) \cap \rho^s(q_{i_s+1}) = \emptyset \\ \forall\, s \in S, s \neq u, i_s < n^s\}$$

Let us now investigate the influence of a control decision in a given state on the makespan. Let x_i be the current state and u the control, i.e., the selected server. Let R^u be the set of resources needed by u to process the next task. Then the time $\tilde{w}$ when u has aquired all needed resources is given by

$$\tilde{w} := \max_{r \in R^u} \{d_r + \max\{\tau(r, o_r, u) - e_r, 0\}\,,$$

where $\tau(r, u, u) := 0$.

The duration Δ until the next control decision is the time when the next server reaches its target position, i.e.,

$$\Delta := \Delta(x_i, u) := \min\bigl\{\min_{s \neq u} w_s, \max\{w_u, \tilde{w}\} + \delta^u\bigr\}\,.$$

So the immediate marginal influence of this decision on the makespan is

$$g_i(x_i, u) := \Delta\,.$$

Since in the terminal state every server s has left all positions including n^s, i.e., all servers are back at the depot according to our convention of "$i_s = n^s$", there is no terminal cost.

Let us now sort out the system dynamics. The next state will be determined by f_i via

$$f_i\bigl((i_s, w_s, o_r, d_r, e_r)_{\substack{s \in S \\ r \in \hat{R}}}, u\bigr) := (i_s{}', w_s{}', o_r{}', d_r{}', e_r{}')_{\substack{s \in S \\ r \in \hat{R}}}$$

with

$$\text{(1)} \qquad i_s{}' := \begin{cases} i_s + 1 & s = u\,, \\ i_s & s \neq u\,, \end{cases}$$

$$\text{(2)} \qquad w_s{}' := \begin{cases} \max\{\tilde{w}, w_u\} + \delta^u - \Delta & s = u\,, \\ \max\{w_s - \Delta, 0\} & s \neq u\,, \end{cases}$$

$$\text{(3)} \qquad o_r{}' := \begin{cases} u & r \in R^u\,, \\ o_r & r \in \hat{R} \setminus R^u\,, \end{cases}$$

$$\text{(4)} \qquad d_r{}' := \begin{cases} \max\{\tilde{w}, w_u\} + \delta^u - \Delta & r \in R^u\,, \\ \max\{0, d_r - \Delta\} & r \in \hat{R} \setminus R^u\,, \end{cases}$$

$$\text{(5)} \qquad e_r{}' := \begin{cases} 0 & r \in R^u\,, \\ \min\{\tau_{\max}(o_r), e_r + \max\{\Delta - d_r, 0\} & r \in \hat{R} \setminus R^u\,. \end{cases}$$

These formulas follow from the fact that Δ time units pass in this state transition. Thus, the will-arrive times w_s of all servers and the delays d_r incurred by all resources will be reduced by Δ, at most to zero. As soon as a resource is not in use any more, i.e., $d_r = 0$, the enabling component e_r starts to measure how much of a possible switching time has passed; it is sufficient to increase it at most to the maximal possible switching time away from the owner $\tau_{\max}(o_r)$. All resources required by u for its next task will be reserved for u. So, the next server who needs them has to wait until u will release them. This finishes the definition of our control problem. The following holds by construction:

Lemma 1. *An optimal control policy for the control problem described above specifies an optimal solution of RSP-T by transcribing the tours and departure times of all servers during the run of the controlled dynamical system.* □

Now that we fully stated our dynamic control problem, it is straight-forward to formulate the standard dynamic programming recursion for it. Let $V_i : X_i \to \mathbb{R}$ be the value function of the ith stage. For each $x_i \in X_i$ the function $V_i(x_i)$ measures the minimal cost needed to reach a terminal state. These are all states with $i_s = q^s_{n^s}$ for every server s. Since there is no terminal cost, we set

$$V_N(x_N) := 0 \text{ for all } x_N \in X_N\,.$$

For all other stages $i = 0, 1, \ldots, N-1$ we use the following standard recursion:

$$V_i(x_i) := \min_{u \in U_i(x_i)} \{g_i(x_i, u) + V_{i+1}\big(f_i(x_i, u)\big)\} \tag{6}$$

and set $\mu_i(x_i)$ to a corresponding minimizer.

Now we can obtain the minimal makespan by evaluating the value function $V_0(x_0)$ with

$$x(0) := (0, 0, \sigma, 0, 0)_{\substack{s\in S \\ r\in\hat{R}}}\,.$$

Lemma 2. *Let M be an upper bound for the makespan of an RSP-T-instance. Then the number of states in the corresponding dynamic control problem is bounded by*

$$\prod_{s\in S} n^s \cdot \big(1+M\big)^{|S|+|\hat{R}|} \cdot \big(\max_{r\in\hat{R}} \max_{s_1,s_2\in S, s_1\neq s_2} \{1+\tau(r,s_1,s_2)\}\big)^{|\hat{R}|} \cdot |S|\,,$$

which is pseudo-polynomial in the input of RSP-T if both $|S|$ and $|\hat{R}|$ are fixed. □

For a constant number of servers, one evaluation of the recursion formula requires only a constant number of steps. Thus, our dynamic programming algorithm runs in pseudo-polynomial time. This implies the following main result of this section, because the constant number of resource assignments can be enumerated.

Theorem 1. *For a fixed number of servers and a fixed total number of resources, RSP-T can be solved in pseudo-polynomial time.* □

It is an open problem whether or not RSP-T with a fixed number of servers and a fixed total number of resources is NP-hard in the ordinary sense.

4. A Branch and Bound Algorithm

Our algorithm is a two stage branch-and-bound approach based on NP-hard subproblems for lower bound computations. We first look at two special cases: the RSP-T where only the scheduling part has to be done and the RSP-s which is a pure routing problem. Both problems will then be used for lower bound calculations in a branch-and-bound algorithm that solves the RSP-J (the subproblem where an assignment of jobs to servers is fixed). known. This first stage will reduce the number of all possible server assignments to a small set of candidates which contains an optimal one. Then we solve the RSP-J for all remaining candidates.

We will use $\ell(\bar{T}^s)$ to denote the (undelayed) length of a tour, i.e., the sum over the distances of all used arcs. For a solution (a scheduled dispatch) $\bar{D} := (\bar{T}^s, \bar{t}^s, \bar{r}^s)_{s\in S}$ we write $\ell(\bar{D})$ for its makespan.

4.1. **The Scheduling Part: RSP-T.** If we fix a tour $\bar{T}^s = (q_1, \dots, q_{n^s})$ for every server, then we obtain a *resource scheduling problem with sequence depended setup times*. We call this special case the RSP-T.

We will now propose a mixed integer model for the RSP-T, which is based on a model original proposed by Tuchscherer et al. [12] for the laser sharing problem, where the laser source was the only resource under consideration. For the given tour $\bar{T}^s = (q_1, \dots, q_{n^s})$ we have to determine a schedule $\bar{t}^s$ and a resource selection $\bar{r}^s$. We will measure the departure times $\bar{t}_i^s$ by the continuous variables $x_i^s \geq 0$. These times will be bounded from below along the tour by

$$x_i^s + \delta^s(q_i^s, q_{i+1}^s) \leq x_{i+1}^s \,.$$

Measuring the makespan using the artificial variable $z \geq 0$ can be done by

$$x_{n^s}^s \leq z, \; \forall\, s \in S \,.$$

The resource selection will be described by the binary variables $u_{s,r}$. We assign each server to exactly one resource of each needed type:

$$\sum_{r \in R} u_{s,r} = 1, \; \forall\, s \in S, R \in \rho^s(\bar{T}^s) \,.$$

When two servers are using the same resource for an arc, we have to decide which one is allowed to use it first and which one has to wait. If two arcs (p_i, p_{i+1}) and (q_j, q_{j+1}) need a resource type R, then these are in conflict if the corresponding servers are both assigned to the same resource r. This will be modeled using binary linear ordering variables y_{p_i,q_j} for all nodes p_i and q_j visited by different servers s_1 and s_2 in their tours $(p_1, \dots, p_n)$ and $(q_1, \dots, q_m)$. Here, $y_{p_i,q_j} = 1$ means that the move (p_i, p_{i+1}) of one server is scheduled to end before the move (q_j, q_{j+1}) of another server that has chosen an identical resource starts. In case of a conflict, the server moves are not allowed to overlap. With M sufficient large, this can be written as

$$x_i^{s_1} + \delta^{s_1}(p_i, p_{i+1}) + \tau(r, s_1, s_2) - x_j^{s_2} \leq M(3 - u_{s_1,r} - u_{s_2,r} - y_{p_i,q_j}) \,.$$

Resource demands on nodes must be handled differently because waiting at a Node v means blocking all assigned resources of $\rho^s(v)$. So, if Server s_1 resides at node p_i and Server s_2 needs a blocked resource to move from q_j to q_{j+1}, then there are only two ways to resolve this conflict. Either s_2 moves from q_j to q_{j+1} before s_1 has reached p_i or after s_1 has left p_i. Because, by our assumption on the resource demand function, Server s_1 needs the conflicting node resources on arcs adjacent to p_i as well, the move of s_1 from p_{i-1} to p_{i+1} over p_i is not allowed to be interrupted by the move q_j to q_{j+1} of s_2:

$$y_{p_{i-1},q_j} = y_{p_i,q_j} \quad \text{whenever } p_i \text{ and } (q_j, q_{j-1}) \text{ are in conflict}.$$

This can be written as

$$\begin{aligned} y_{p_{i-1},q_j} - y_{p_i,q_j} &\leq 2 - u_{s_1,r} - u_{s_2,r} \\ y_{p_i,q_j} - y_{p_{i-1},q_j} &\leq u_{s_1,r} + u_{s_2,r} - 2 \,. \end{aligned}$$

If a conflict for a node resource occurs at a depot, i.e., $i = 1$ or $i = n$, then the move (q_j, q_{j+1}) has to be after the move (p_1, p_2) and before the move (p_{n-1}, p_n):

$$\begin{aligned} 1 - y_{p_1,q_j} &\leq 2 - u_{s_1,r} - u_{s_2,r} \\ 1 - y_{q_j,p_{n-1}} &\leq 2 - u_{s_1,r} - u_{s_2,r} \,. \end{aligned}$$

We now summarize the whole model:

Problem 1 (RSP-T).

$$\min z \tag{7}$$

subject to

$$x^s_{n^s} \le z \qquad \forall\, s \in S \tag{8}$$

$$x^s_i + \delta^s(q_i, q_{i+1}) \le x^s_{i+1} \qquad \forall\, s \in S, i = 1, \dots, n^s - 1 \tag{9}$$

$$\begin{aligned} & x^{s_1}_i + \delta^{s_1}(p_i, p_{i+1}) \\ & + \tau(r, s_1, s_2) - x_{s_2,j} \le M(3 - u_{s_1,r} \\ & \qquad - u_{r,s_2} - y_{p_i,q_j}) \end{aligned} \qquad \begin{aligned} & \forall\, s_1, s_2 \in S, s_1 \ne s_2, \\ & i = 1, \dots, n^{s_1} - 1, \\ & j = 1, \dots, n^{s_2} - 1, \\ & R \in \rho^{s_1}(p_i, p_{i+1}) \cap \rho^{s_2}(q_j, q_{j+1}), \\ & r \in R \end{aligned} \tag{10}$$

$$y_{p_{i-1},q_j} - y_{p_i,q_j} \le 2 - u_{s_1,r} - u_{s_2,r} \qquad \begin{aligned} & \forall\, s_1, s_2 \in S, s_1 \ne s_2 \\ & i = 2, \dots, n^{s_1} - 1, \\ & j = 1, \dots, n^{s_2} - 1 \\ & R \in \rho^{s_1}(p_i) \cap \rho^{s_2}(q_j, q_{j+1}), \\ & r \in R \end{aligned} \tag{11}$$

$$y_{p_i,q_j} - y_{p_{i-1},q_j} \le 2 - u_{s_1,r} - u_{s_2,r} \qquad \begin{aligned} & \forall\, s_1, s_2 \in S, s_1 \ne s_2 \\ & i = 2, \dots, n^{s_1} - 1, \\ & j = 1, \dots, n^{s_2} - 1 \\ & R \in \rho^{s_1}(p_i) \cap \rho^{s_2}(q_j, q_{j+1}), \\ & r \in R \end{aligned} \tag{12}$$

$$1 - y_{p_i,q_j} \le 2 - u_{s_1,r} - u_{s_2,r} \qquad \begin{aligned} & \forall\, s_1, s_2 \in S, s_1 \ne s_2 \\ & i = 1, j = 1, \dots, n^{s_2} - 1 \\ & R \in \rho^{s_1}(d^{s_1}) \cap \rho^{s_2}(q_j, q_{j+1}), \\ & r \in R \end{aligned} \tag{13}$$

$$1 - y_{q_j,p_i} \le 2 - u_{s_1,r} - u_{s_2,r} \qquad \begin{aligned} & \forall\, s_1, s_2 \in S, s_1 \ne s_2 \\ & i = n^{s_1} - 1, j = 1, \dots, n^{s_2} - 1 \\ & R \in \rho^{s_1}(d^{s_1}) \cap \rho^{s_2}(q_j, q_{j+1}), \\ & r \in R \end{aligned} \tag{14}$$

$$y_{p,q} + y_{q,p} = 1 \qquad \begin{aligned} & \forall\, s_1, s_2 \in S, s_1 \ne s_2, \\ & p \in V^{s_1}(\bar{T}^{s_1}), q \in V^{s_2}(\bar{T}^{s_2}) \end{aligned} \tag{15}$$

$$\sum_{r \in R} u_{r,s} = 1 \qquad \forall\, s \in S, R \in \rho^s(\bar{T}^s) \tag{16}$$

$$x^s_i \ge 0 \qquad \forall\, s \in S, i = 1, \dots, n^s \tag{17}$$

$$z \ge 0 \tag{18}$$

$$y_{p,q} \in \{0, 1\} \qquad \begin{aligned} & \forall\, s_1, s_2 \in S, s_1 \ne s_2, \\ & p \in V^{s_1}(\bar{T}^{s_1}), q \in V^{s_2}(\bar{T}^{s_2}) \end{aligned} \tag{19}$$

$$u_{s,r} \in \{0,1\} \qquad \forall\, s \in S, R \in \rho^s(\bar{T}^s), r \in R \tag{20}$$

We will make use of an exact solver for the RSP-T in the following way: Given a partial tour $\bar{T}^s$ for every Server s, i.e., a tour not ending in d^s, and a lower bound l^s for the time needed to complete $\bar{T}^s$, we can replace $\delta^s(q_{n^s-1}, d^s)$ by l^s in (9) and solve the RSP-T for the partial tours. This way we end up with a lower bound for all feasible solutions to the RSP-T that start with $\bar{T}^s$.

We will write $\bar{D}_{\text{RSP-T}}(\bar{T}, l) := \text{RSP-T}(\bar{T}, l)$ with $\bar{T} := (\bar{T}^s)_{s \in S}$ for a call to an exact solver solving Problem 1 that stores an optimal scheduled dispatch in $\bar{D}_{\text{RSP-T}}(\bar{T}, l)$. If the problem is infeasible, then $\bar{D}_{\text{RSP-T}}(\bar{T}, l) := \emptyset$ with $\ell(\emptyset) := \infty$.

4.2. **The Routing Part: RSP-s.**

If there is only one server available, then no resource constraint plays a role, and we end up with a pure routing problem. In this section we will discuss this special case in more detail.

Server s has to find a route in $G := G^s$ with minimal length. The tour first starts at d^s, then processes all jobs (either in direction $j_a \to j_b$ or vise versa), and finally returns home. Thus, Server s has to visit every position exactly once. This can be modeled as an *asymmetric traveling salesman problem* with the additional constraint that for every job $j \in J$ one of the two arcs (j_a, j_b) or (j_b, j_a) must be part of the solution.

Therefore, we can model the RSP-s by the following ILP with $c_{v,w} := \delta^s(v,w)$:

Problem 2 (RSP-s).

$$\min \sum_{(v,w) \in A} c_{v,w} x_{v,w}$$

subject to

$$\sum_{(v,w) \in A} x_{v,w} = 1 \qquad \forall\, v \in V \tag{21}$$

$$\sum_{(v,w) \in A} x_{v,w} = 1 \qquad \forall\, w \in V \tag{22}$$

$$\sum_{(v,w) \in A \cap (U \times U)} x_{v,w} \leq |U| - 1 \qquad U \subset V, 2 \leq |U| \leq |V| - 2 \tag{23}$$

$$x_{j_a,j_b} + x_{j_b,j_a} = 1 \qquad \forall\, j \in J \tag{24}$$

$$x_{v,w} \in \{0,1\} \qquad \forall\, (v,w) \in A \tag{25}$$

Constraints (21)–(22) are the degree constraints of the standard ATSP model, and (23) are subtour elimination constraints. We added (24) to ensure that exactly one of the job arcs is processed.

Lemma 3. *Every valid inequality for the ATSP polytope is also valid for the RSP-s.* □

Solving the RSP-s can be done by branch-and-cut, where every valid ATSP inequality can be used as a cutting plane. See [9] for a polyhedral approach to solve a standard ATSP. There is also a way to transform the RSP-s into a standard ATSP: For each $j \in J$ introduce another city j_c between j_a and j_b with $c_{j_a,j_c} + c_{j_c,j_b} = c_{j_a,j_b}, c_{j_b,j_c} + c_{j_c,j_a} = c_{j_b,j_a}$ and $c_{j_c,v} := c_{v,j_c} := M$ for a large constant M otherwise. Now any ATSP solver can be used to solve the RSP-s. It is also possible to transform the constructed ATSP to a symmetric TSP and use `Concorde` [1]. Due to technical limitation of the Concorde code, e.g., all costs have to be integer, we avoided this (in contrast to an earlier version of our algorithm).

We will use solutions to the RSP-s in the following way: Consider a set of jobs $\tilde{J} \subseteq J$ and the current position p of Server s. Set $c_{d^s,v} := \delta^s(p,v)$ for all $v \in V$. Then, a solution to the correponding RSP-s evaluates how long s needs at least to process all jobs of $\tilde{J}$ starting from p. We write $\bar{T}^s_{\text{RSP-s}}(\tilde{J},p) := \text{RSP-s}^s(\tilde{J},p)$ for a call to an exact algorithm solving the corresponding RSP-s that stores the resulting optimal tour in $\bar{T}^s_{\text{RSP-s}}(\tilde{J},p)$.

4.3. **Combining routing and scheduling for fixed assignment of jobs : RSP-J.** Let $\bar{J} : S \to 2^J$ be a given assignment of jobs to servers. Then we can assume $J^s = \bar{J}^s$, i.e., for each server only the jobs assigned to it are feasible for it. In the absence of resource sharing, the tours of all servers can be computed independently, and thus we only have to solve $|S|$ problems of type RSP-s. If resources have to be taken into account, then this still supplies us with a lower bound:

Lemma 4. *Let $\bar{T}^s_{RSP\text{-}s}(J^s,d^s)$ be an optimal solution for the RSP-s for each server s and $\bar{D} := (\bar{T}^s,\bar{t}^s,\bar{r}^s)_{s\in S}$ be any feasible solution to the RSP-J, then we have*

$$\square \qquad \max_{s\in S} \ell\big(\bar{T}^s_{RSP\text{-}s}(J^s,d^s)\big) \le \ell(\bar{D}).$$

Our branch-and-bound approach now works as follows: Assume we are given a partial tour $\bar{T}^s = (q_1,\dots,q^s)$ for each server. The sets of jobs visited by server s is denoted by $J(\bar{T}^s)$. Then, no tour starting with $\bar{T}^s$ visiting all remaining jobs $K^s := J^s \setminus J(\bar{T}^s)$ can finish earlier than the concatenation of $\bar{T}^s$ and $\bar{T}^s_{\text{RSP-s}}(K^s,q^s)$. Hence,

$$\max_{s\in S}\Big(\ell(\bar{T}^s) + \ell\big(\bar{T}^s_{\text{RSP-s}}(K^s,q^s)\big)\Big) \tag{26}$$

is a valid lower bound. In order to improve this bound further we make use of the resource demands of the partial tours. This can be done by using the following fact: Every tour for s starting with $\bar{T}^s$ needs at least $\ell(\bar{T}^s_{\text{RSP-s}}(K^s,q^s))$ time units to move from position q^s back to d^s. With $l^s := \ell(\bar{T}^s_{\text{RSP-s}}(K^s,q^s))$ we can improve our bound solving an RSP-T problem. Recall that $\ell\big(\bar{D}_{\text{RSP-T}}(\bar{T},l)\big)$ is the minimal makespan of a scheduled dispatch starting with partial tours $\bar{T} = (\bar{T}^s)_{s\in S}$, when the driving times from the last positions of $\bar{T}$ to the depots are given by $l = (l^s)_{s\in S}$.

Lemma 5. *Let $\bar{T} := (\bar{T}^s)_{s\in S}$ be a set of partial tours and $\bar{D}$ be a feasible scheduled dispatch whose tour set starts with $\bar{T}$. Let $l := (l^s)_{s\in S}$, $l^s := \ell(\bar{T}^s_{RSP\text{-}s}(K^s,q^s))$, where q^s is the last position of $\bar{T}^s$. Then we have*

$$\square \qquad \max_{s\in S}\big(\ell(\bar{T}^s) + l^s\big) \le \ell\big(\bar{D}_{RSP\text{-}T}(\bar{T},l)\big) \le \ell(\bar{D}).$$

A node in our branch-and-bound tree corresponds to a partial tour for every server. In the branching step we have to choose a server that has unserved jobs. We then create a child node for every $j \in K^s$ and every possible start position $q \in \{j_a, j_b\}$. A leaf corresponds of a full tour for each server. Finally, we evaluate optimal schedulings in the leaves by solving the corresponding RSP-T problems.

We will write N for a node and $\bar{T}(N) = (\bar{T}^s)_{s\in S}$ for the partial tours associated with N. Moreover, $\lambda(N)$ is the lower bound for Node N according to Lemma 5. The set of jobs served in a routing $\bar{T}$ is denoted by $J(\bar{T})$. Algorithm 1 summarizes the method.

Lemma 6. *If $\bar{J} : S \to 2^J$ corresponds to the job-server assignment of an optimal solution to the* RSP*, then Algorithm 1 will find an optimal solution of the* RSP.

Remark 5. Let $\bar{T}^s_{\text{concat}}$ be the concatenation of $\bar{T}^s$ and $\bar{T}^s_{\text{RSP-s}}(K^s,q^s)$. Solving the RSP-T with a tour set $\bar{T}_{\text{concat}} := (\bar{T}^s_{\text{concat}})_{s\in S}$ yields a primal feasible solution to the RSP-J in case that the RSP-T is not infeasible. This primal heuristic can be integrated into Algorithm 1 to obtain an upper bound μ in every node.

Algorithm 1 Combinatorial Branch-and-Bound for the RSP-J

```
Require: Data of the RSP-J
Ensure: $(\bar{T}^s_{\text{OPT}}, \bar{t}^s_{\text{OPT}}, \bar{r}^s_{\text{OPT}})_{s\in S}$ is an optimal solution to the RSP-J.
  $\bar{T}^s \leftarrow (d^s), s \in S$
  $\bar{T}(N) \leftarrow (\bar{T}^s)_{s\in S}$
  add $N$ to set of nodes
  while set of nodes not empty do
    pick $N$ from set of nodes
    for all $s \in S$ do
      $K^s \leftarrow J^s \setminus J(\bar{T}^s)$
      $q^s \leftarrow \text{endpos}(\bar{T}^s)$
      $\bar{T}^s_{\text{RSP-s}}(K^s, q^s) \leftarrow \text{RSP-s}^s(K^s, q^s)$
      $l^s \leftarrow \ell\big(\bar{T}^s_{\text{RSP-s}}(K^s, q^s)\big)$
    end for
    $l \leftarrow (l^s)_{s\in S}$
    $\bar{D}_{\text{RSP-T}}(\bar{T}(N), l) \leftarrow \text{RSP-T}(\bar{T}, l)$
    $\lambda(N) \leftarrow \ell\big(\bar{D}_{\text{RSP-T}}(\bar{T}(N), l)\big)$
    if $\lambda(N) < \mu$ then
      if $J(\bar{T}^s) = J^s \;\forall\, s \in S$ then
        for all $s \in S$ do
          append $d^s$ to $\bar{T}^s$
          $\bar{T}^s_{\text{OPT}} \leftarrow \bar{T}^s$
          set $\bar{r}^s_{\text{OPT}}, \bar{t}^s_{\text{OPT}}$ according to $\bar{D}_{\text{RSP-T}}(\bar{T}(N), l)$
        end for
      else
        select $\hat{s} \in S$ with $K^{\hat{s}} \neq \emptyset$
        for all $j \in K_{\hat{s}}$ do
          create node $N_1$ with $(j_a, j_b)$ added to $\bar{T}^{\hat{s}}$
          create node $N_2$ with $(j_b, j_a)$ added to $\bar{T}^{\hat{s}}$
          add $N_1$ and $N_2$ to set of nodes
        end for
      end if
    end if
  end while
  return $(\bar{T}^s_{\text{OPT}}, \bar{t}^s_{\text{OPT}}, \bar{r}^s_{\text{OPT}})_{s\in S}$
```

4.4. **Finding an optimal server assignment: Solving the** RSP**.** Due to Lemma 6, we only need to find out the server assignment $\bar{J}$ of an optimal solution. Then we can apply Algorithm 1 to obtain an optimal solution to the RSP. Simple enumeration would mean $\prod_{j\in J} |\{s \mid j \in J^s\}|$ calls of Algorithm 1.

In this subsection we will propose another branch-and-bound algorithm: it reduces this large number to a usually small set of candidates. For this purpose we need an additional assumption: The distances need to satisfy the triangle inequality.

For a given assignment $\bar{J} : S \to 2^J$ we can use Lemma 4 to produce a lower bound. In order to be able to branch on incremental assignment decisions, we will now extend this bound to partial assignments. Each possibly partial assignment is completely determined by the sets $\bar{J}^s$ of jobs assigned to Server s for $s \in S$.

Algorithm 2 Candidate Selection for the RSP

```
Require: Data of the RSP
Ensure: The set $\Omega$ contains an optimal server assignment $\big(\bar{J}^s\big)_{s\in S}$ of the RSP.
  $\mu \leftarrow \infty$
  $\Omega \leftarrow \emptyset$
  for all $s \in S$ do
    $\bar{J}^s(N) \leftarrow \emptyset$
  end for
  add $N$ to set of nodes
  while set of nodes not empty do
    pick $N$ from set of nodes
    for all $s \in S$ do
      $\bar{T}^s_{\text{RSP-s}}(\bar{J}^s(N), d^s) \leftarrow \text{RSP-s}^s(\bar{J}^s(N), d^s)$
    end for
    $\lambda(N) \leftarrow \max_{s\in S} \ell\big(\bar{T}^s_{\text{RSP-s}}(\bar{J}^s(N), d^s)\big)$
    if $\lambda(N) < \mu$ then
      if $\cup_{s\in S}\bar{J}^s(N) = J$ then
        add $(\bar{J}^s(N))_{s\in S}$ to $\Omega$
        for all $s \in S$ do
          $q^s \leftarrow \text{endpos}(\bar{T}^s_{\text{RSP-s}}(\bar{J}^s(N), d^s))$
          $l^s \leftarrow \delta^s(q^s, d^s)$
        end for
        $T \leftarrow (\bar{T}^s_{\text{RSP-s}}(\bar{J}^s(N), d^s))_{s\in S}$
        $l \leftarrow (l^s)_{s\in S}$
        $\bar{D}_{\text{RSP-T}}(T, l) \leftarrow \text{RSP-T}(T, l)$
        if $\ell\big(\bar{D}_{\text{RSP-T}}(T, l)\big) < \mu$ then
          $\mu \leftarrow \ell\big(\bar{D}_{\text{RSP-T}}(T, l)\big)$
        end if
      else
        select $j \in J \setminus \cup_{s\in S}\bar{J}^s(N)$
        for all $s \in S$ do
          if $s \in J^s$ then
            create node $N$ with $j$ added to $\bar{J}^s(N)$
            add $N$ to set of nodes
          end if
        end for
      end if
    end if
  end while
  return $\Omega$
```

Proposition 3. *Let $J_1 \subseteq J_2 \subseteq J$ and $\bar{T}^s_i := \bar{T}^s_{RSP\text{-}s}(J_i, d^s)$, $i = 1,2$, be optimal solutions the corresponding RSP-s. Then we have*

$$\ell(\bar{T}^s_1) \leq \ell(\bar{T}^s_2)\,.$$

Proof. Let $\bar{T}^s$ be obtained by deleting all $j \in J_2 \setminus J_1$ from $\bar{T}^s_2$ short-cutting the tour. Then, by the triangle inequality, we have $\ell(\bar{T}^s) \leq \ell(\bar{T}^s_2)$. Since $\bar{T}^s$ is feasible for the RSP-s with respect to J_1, the optimality of $\bar{T}^s_1$ yields $\ell(\bar{T}^s_1) \leq \bar{T}^s$. □ □

Due to Proposition 3 and Lemma 4 we can use the following lower bound

$$\lambda(\bar{J}) := \max_{s\in S} \ell\big(\bar{T}^s_{\text{RSP-s}}(\bar{J}^s, d^s)\big)\,. \tag{27}$$

A node in the branch-and-bound algorithm now corresponds to a partial assignment. We use the idea of Remark 5 to get a primal solution whenever we reach a leaf; this may improve our current upper bound. Every leaf whose lower bound is below the current best upper bound is a *candidate* for an optimal assignment.

The method for candidate collection is shown in Algorithm 2. We will write $\bar{J}(N)$ for the partial assignment of node N, encoded by the sets $\bar{J}^s(N) \subseteq J^s$, i.e., the sets of jobs assigned to Server s in Node N for $s \in S$.

Lemma 7. *When the* RSP *is feasible, then the set* Ω *returned by Algorithm 2 contains the server assignment of an optimal solution.*

The top-level method solving the RSP is shown in Algorithm 3.

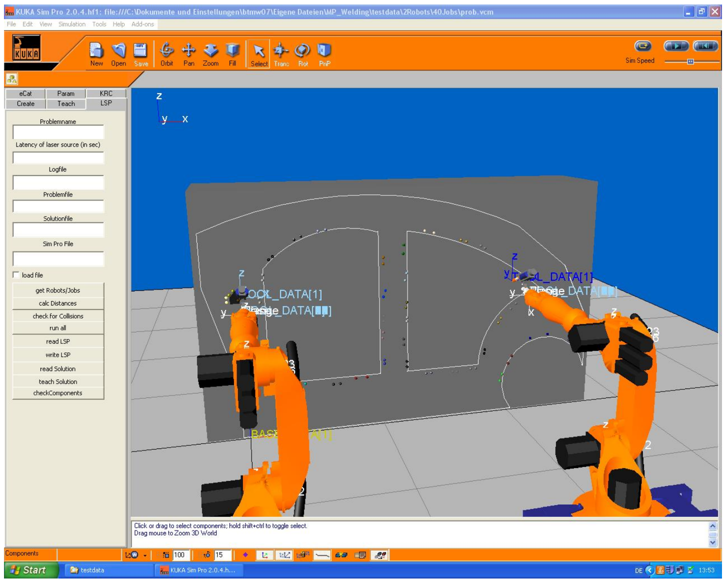
Algorithm 3 Combinatorial Branch-and-Bound (CBB) for the RSP

```
Require: Data of the RSP
Ensure: $(\bar{T}^s_{\text{OPT}}, \bar{t}^s_{\text{OPT}}, \bar{r}^s_{\text{OPT}})_{s\in S}$ is an optimal solution to the RSP.
  $\mu \leftarrow \infty$
  get $\Omega$ by Algorithm 2
  for all $\bar{J} \in \Omega$ do
    get $\bar{D} := (\bar{T}^s, \bar{r}^s, \bar{t}^s)_{s\in S}$ by Algorithm 1
    if $\ell(\bar{D}) < \mu$ then
      for all $s \in S$ do
        $\bar{T}^s_{\text{OPT}} \leftarrow \bar{T}^s$
        $\bar{r}^s_{\text{OPT}} \leftarrow \bar{r}^s$
        $\bar{t}^s_{\text{OPT}} \leftarrow \bar{t}^s$
      end for
    end if
  end for
  return $(\bar{T}^s_{\text{OPT}}, \bar{t}^s_{\text{OPT}}, \bar{r}^s_{\text{OPT}})_{s\in S}$
```

5. Computational Results

In this section, we present comprehensive computational results achieved by our algorithm CBB for the RSP with data from the LSP problem and compare them to other reasonable approaches based on running cplex on global MILP models.

It is important to note that our algorithm – though formulated for the general RSP – was designed with the special properties of industrial-type LSP instances in mind. This means that the results in the sequel and their interpretation is only valid for RSPs with similar characteristics. The most important one is: Since we are interested in increasing the productivity of the most expensive resources (the laser sources) by using fewer of them, the problem is only interesting if the productivity of this resource is not satisfactory. Moreover, since the total processing time is fixed and the solution with unshared resources almost uses this processing time completely, a gain by sharing the laser sources among robots is only possible if sharing does not necessarily increase the makespan too much. But this means, that the bound from routing with resource sharing ignored must already be quite tight.


KUKA Sim Pro 2.0.4.hf1: file:///C:\Dokumente und Einstellungen\btmw07\Eigene Dateien\MP_Welding\testdata\2Robots\40Jobs\prob.vcm

Click or drag to select components; hold shift+ctrl to toggle select.
Drag mouse to Zoom 3D World

FIGURE 1. Data file with two robots

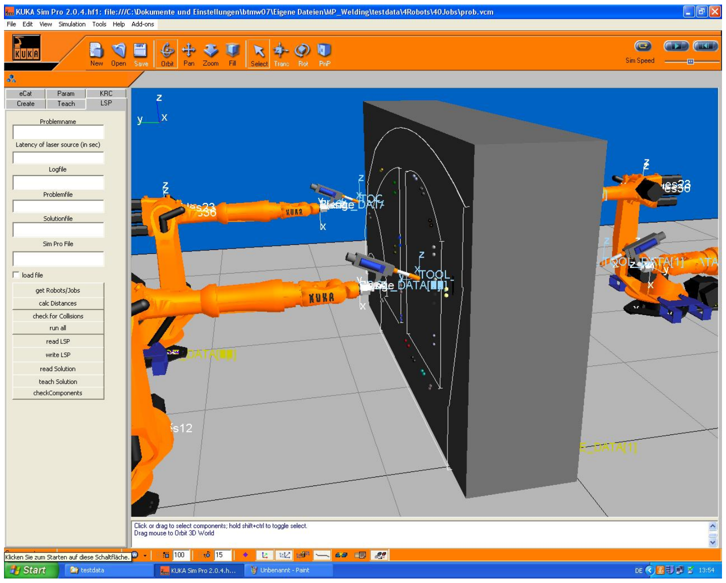
Since we know of no other algorithms for the LSP, we will compare our method to standard MILP approaches. Historically, collisions where not part of the LSP [12, 19, 17]. All mixed integer models are therefore described without collision avoidance. For each model, we will discuss whether it can be extended to contain collision avoidance. Our own algorithm was tested with and without collision avoidance whereever sensible.

5.1. **Data Generation.** We created two simulation models. Figure 1 shows the instance for two robots with 40 jobs and Figure 2 the one for four robots and 40 jobs. Smaller instances where obtained by dropping jobs one by one. In this way any instance with n jobs includes all jobs of the instance with $n-1$ jobs. The job end positions are marked with tiny spheres, same color indicates same job.

How long Robot s needs to move from Position p to Position q depends on the angle settings at the positions and on the path planning. A position is usually given by six coordinates (x,y,z,a,b,c), where (x,y,z) are the cartesian coordinates of the top of the welding gun, and (a,b,c) specifies in which angle the welding beam hits the component.

To overcome the problem of non-unique distances, we fixed a unique, technically plausible angle setting for each position of the robots. For every measurement, we taught the robot moves using these settings. We used linear path planning, which ensures that no collision between the robots and the component will occur. More complex path planning may be employed, but due the short distances, the speed up will usualy be neglectable.

We used KuKa's built-in tool for collision checking. This works by specifiying two sets of geometry components A and B and evaluating their intersection in the simulation world. The tool will check only whether there are two components $a \in A, b \in B$ colliding in a particular position. In order to check a line-line collision $(r, p_1, p_2, s, q_1, q_2)$, both lines (p_1, p_2) and (q_1, q_2) were be discretized. Then we moved Robot r to every position p and Robot s to every position q of their path discretization and checked for a collision. Line-point collisions were checked in the same way.


KUKA Sim Pro 2.0.4.hf1: file:///C:\Dokumente und Einstellungen\btmw07\Eigene Dateien\MP_Welding\testdata\4Robots\40Jobs\prob.vcm

Click or drag to select components; hold shift+ctrl to toggle select.
Drag mouse to Orbit 3D World

FIGURE 2. Data file with four robots

5.2. **Mixed Integer Linear Models.** We will now describe the mixed integer linear models we use for comparision. This is not so much meant as a competition between proposed solution methods. We rather want to show that various types of global polyhedral models may cause problems because the scheduling part and the makespan objective make LP relaxations weak. Optimal solutions to combinatorial subproblems, however, yield much tighter bounds in significantly less time.

Our first MILP model is a model based on linear ordering variables and represents compact models with variables for atomic decisions and consequently more complicated side constraints. The makespan objective is evaluated using a continuous variable z. For every job j the continuos variable x_j measures the start time of the welding task j. The binary variables $m_{j,p}$ indicate whether processing j is started at position $p \in \{j_a, j_b\}$. We will write q as the corresponding end position, i.e., if $p = j_a$ then $q = j_b$, and vice versa. The job-robot assignment is described by $v_{j,s} \in \{0,1\}$, the robot-laser source assignment by $u_{s,l}$. For every pair j_1, j_2 of jobs the linear order variable $y_{j_1,j_2} \in \{0,1\}$ denotes whether j_1 is processed before j_2.

Problem 3 (Linear Ordering (lo)).

$$\min z$$

subject to

$$z - x_j - \delta^s(p,q) - \delta^s(q,d^s) \geq -M(2 - m_{j,p} - v_{j,s}) \qquad \forall\, s \in S, j \in J, \; p \in \{j_a, j_b\} \tag{28}$$

$$x_j - \delta^s(d^s,p) \geq -M(2 - m_{j,p} - v_{j,s}) \qquad \forall\, s \in S, j \in J, \; p \in \{j_a, j_b\} \tag{29}$$

$$\begin{aligned} x_{j_1} - x_{j_2} + \delta^s(p_1,q_1) + \delta^s(q_1,p_2) \leq M(5 - m_{j_1,p_1} - m_{j_2,p_2} \\ - v_{j_1,s} - v_{j_1,s} - y_{j_1,j_2}) \end{aligned} \qquad \forall\, s \in S, \forall\, j_1, j_2 \in J, \; p_i \in \{j_{ia}, j_{ib}\}, i = 1,2, \tag{30}$$

$$x_{j_1} - x_{j_2} + \sum_{p_1 \in \{j_{1a}, j_{1b}\}} m_{j_1,p_1} \delta^s(p_1,q_1) + \tau^l \le M(5 - v_{j_1,s_1} - v_{j_2,s_2} - u_{l,s_1} - u_{l,s_2} - y_{j_1,j_2}) \quad \forall\, l \in L, \forall\, s_1,s_2 \in S,\ s_1 \neq s_2,\ \forall\, j_1,j_2 \in J, j_1 \neq j_2 \tag{31}$$

$$\sum_{l \in L} u_{l,s} = 1 \quad \forall\, s \in S \tag{32}$$

$$\sum_{s \in S: j \in J^s} v_{j,s} = 1 \quad \forall\, j \in J \tag{33}$$

$$m_{j,j_a} + m_{j,j_b} = 1 \quad \forall\, j \in J \tag{34}$$

$$y_{j_1,j_2} + y_{j_2,j_1} = 1 \quad \forall\, j_1,j_2 \in J, j_1 \neq j_2 \tag{35}$$

$$y_{j_1,j_2} \in \{0,1\} \quad \forall\, j_1,j_2 \in J, j_1 \neq j_2 \tag{36}$$

$$m_{j,p} \in \{0,1\} \quad \forall\, j \in J, p \in \{j_a, j_b\} \tag{37}$$

$$x_j \ge 0 \quad \forall\, j \in J \tag{38}$$

$$z \ge 0 \tag{39}$$

Constraint (28) measures the makespan, (29) ensures that the first job is not processed before the robot can reach it, (30) measures the start times along the welding paths, and (31) ensures that every laser source is used for at most one job at a time.

Nearly all constraints are modeled by using big-Ms. This tends to give poor bounds in the LP relaxation. If we aggregate the information of $m_{j,p}$ and $v_{j,s}$ into the variable $w_{s,j,p} \in \{0,1\}$ we can get rid of th big-Ms in (28) and (29), resulting in the following variant of the linear order model above.

Problem 4 (Linear Ordering Variant (lov)).

$$\min z$$

subject to

$$x_j + \sum_{s \in S} \sum_{p \in \{j_a, j_b\}} \big(\delta^s(p,q) + \delta^s(q,d^s)\big) w_{s,j,p} \le z \quad \forall\, s \in S, j \in J \tag{40}$$

$$\sum_{s \in S} \sum_{p \in \{j_a, j_b\}} \delta^s(d^s,p) w_{s,j,p} \le x_j \quad \forall\, s \in S, j \in J \tag{41}$$

$$x_{j_1} - x_{j_2} + \delta^s(p_1,q_1) + \delta^s(q_1,p_2) \le M(3 - w_{s,j_1,p_1} - w_{s,j_2,p_2} - y_{j_1,j_2}) \quad \forall\, s \in S, \forall\, j_1,j_2 \in J,\ p_i \in \{j_{ia}, j_{ib}\}, i = 1,2 \tag{42}$$

$$x_{j_1} - x_{j_2} + \sum_{p_1 \in \{j_{1a}, j_{1b}\}} \big(\delta^{s_1}(p_1,q_1) + \tau^l\big) w_{s,j_1,p_1} \le M(4 - w_{s_2,j_2,j_{2a}} - w_{s_2,j_2,j_{2b}} - u_{l,s_1} - u_{l,s_2} - y_{i,j}) \quad \forall\, l \in L, \forall\, s_1,s_2 \in S,\ s_1 \neq s_2,\ \forall\, j_1,j_2 \in J, j_1 \neq j_2 \tag{43}$$

$$\sum_{s \in S: j \in J^s} \sum_{p \in \{j_a, j_b\}} w_{s,j,p} = 1 \quad \forall\, j \in J \tag{44}$$

$$y_{j_1,j_2} + y_{j_2,j_1} = 1 \quad \forall\, j_1,j_2 \in J, j_1 \neq j_2 \tag{45}$$

$$y_{j_1,j_2} \in \{0,1\} \quad \forall\, j_1,j_2 \in J, j_1 \neq j_2 \tag{46}$$

$$w_{s,j,p} \in \{0,1\} \quad \forall\, s \in S, j \in J,$$

$$p \in \{j_a, j_b\} \tag{47}$$

$$x_j \geq 0 \qquad \forall\, j \in J \tag{48}$$

$$z \geq 0 \tag{49}$$

The following is straight-forward.

Lemma 8. *Let μ be an upper bound for the makespan; moreover, let*

$$d := \max\{\ \delta^s(p,q) \mid p,q \in Q \text{ with } \delta^s(p,q) < \infty, s \in S\ \}\,.$$

Then M can be chosen as $\mu + 2d + \max_{l \in L} \tau^l$. □

Remark 6. Both models need the triangle inequality for the actually needed times between positions. While this is valid for our distance function (undelayed), care must be taken in the presence of collision avoidance. A line-line collision can be resolved in two ways. Either wait until the collision resource is available again or choose a different next position. Choosing the detour may be faster because of less waiting time. This way the triangle inequality may be violated.

A third model uses a large flow network for its decisions, modeling most of the logic in its structure. It represents models using expanded graphs to reflect logical structures. The model is based on the following idea. Instead of x_j we use $x_{l,i}$ measuring the start time of the i-th job processed using Laser Source l. Then we have $x_{l,i} \leq x_{l,i+1}$ by construction, which gives us the possibility to avoid big-Ms alltogether. For this purpose let $G^{s,l} := G(V^{s,l}, A^{s,l})$ be a digraph with nodes

$$V^{s,l} := \{(j,p,i), j \in J, p \in \{j_a, j_b\}, i = 1,\dots,|J|\} \cup \{d^s\}\,.$$

A tour of Robot s can now be viewed as a tour in $G^{s,l}$. Visiting Node (j,p,i) is interpreted as processing Job j starting at Position p as the i-th job on Laser Source l. Each robot can choose one laser source to work with. This is modeled by choosing one graph $G^{s,l}$ for each robot.

The orders in which the robots process their jobs preserves the orders of the jobs on the assigned laser sources. So, the arc sets are

$$\begin{aligned} A^{s,l} := \{(v,w) \in V^{s,l} :& v = d^s \vee w = d^s \\ & \vee v = (j_1,p_1,i_1), w = (j_2,p_2,i_2) \wedge i_1 < i_2\} \end{aligned}$$

Besides the makespan variable z and the already mentioned time variables $x_{l,i}$ we have flow variables $y_a \in \{0,1\}, a \in A^{l,s}$. Let $I := \{1,\dots,|J|\}$.

Problem 5 (Flow Model (fm)).

$$\min z$$

subject to

$$x_{l,i} + \sum_{\substack{s \in S \\ ((j,p,i),d^s) \in A^{s,l}}} \big(\delta^s(p,q) + \delta^s(q,d^s)\big) y^{s,l}_{((j,p,i),d^s)} \leq z \qquad \forall\, l \in L, i \in I \tag{50}$$

$$\sum_{\substack{s \in S \\ (d^s,(j,p,i)) \in A^{s,l}}} \delta^s(d^s,(j,p,i)) y^{s,l}_{(d^s,(j,p,i))} \leq x_{l,i} \qquad \forall\, l \in L, i \in I \tag{51}$$

$$x_{l,i_1} + \sum_{\substack{s \in S \\ ((j_1,p_1,i_1),(j_2,p_2,i_2)) \in A^{s,l}}} \big(\delta^s(p_1,q_1) + \delta^s(q_1,p_2)\big) y^{s,l}_{((j_1,p_1,i_1),(j_2,p_2,i_2))} \leq x_{l,i_2} \qquad \forall\, l \in L, \; i_1, i_2 \in I, i_1 < i_2 \tag{52}$$

$$x_{l,i} + \sum_{\substack{s\in S\\ ((j_1,p_1,i_1),(j_2,p_2,i_2))\in A^{s,l}\\ i_2>i_1+1}} (\delta^s(p_1,q_1)+\tau^l) y^{r,l}_{((j_1,p_1,i_1),(j_2,p_2,i_2))} + \sum_{\substack{s\in S\\ ((j,p,i),d^s)\in A^{r,l}}} (\delta^s(p,q)+\tau^l) y^{s,l}_{((j,p,i),d^s)} \le x_{l,i+1} \qquad \forall\, l\in L, i\in I,\ i<|J| \tag{53}$$

$$\sum_{l\in L}\sum_{(d^s,v)\in A^{s,l}} y^{s,l}_{(d^s,v)} \le 1 \qquad \forall\, s\in S \tag{54}$$

$$\sum_{(w,v)\in A^{s,l}} y^{s,l}_{(v,w)} - \sum_{(v,w)\in A^{s,l}} y^{s,l}_{(w,v)} = 0 \qquad \forall\, l\in L, s\in S,\ v\in V^{s,l} \tag{55}$$

$$\sum_{\substack{l\in L\\ s\in S}}\sum_{(v,(j,p,i))\in A^{l,s}} y^{s,l}_{(v,(j,p,i))} = 1 \qquad \forall\, j\in J \tag{56}$$

$$\sum_{s\in S}\sum_{(v,(j,p,i))\in A^{s,l}} y^{s,l}_{(v,(j,p,i))} \le 1 \qquad \forall\, l\in L, i\in I \tag{57}$$

$$y^{s,l}_a \in \{0,1\} \qquad \forall\, l\in L, s\in S,\ a\in A^{s,l} \tag{58}$$

$$x_{l,i} \ge 0 \qquad \forall\, l\in L, i\in I \tag{59}$$

$$z \ge 0 \tag{60}$$

The makespan is measured in (50), and (51) evaluates the start time of the first job of each robot. Constraint (52) evaluates the start times along the robot paths; and (53) calculates the waiting times for the laser sources. The assignment of robots to laser sources is done by the robot packing constraint (54). The laser packing constraint (57) ensures that only one job is welding at a time using laser source l. The partitioning constraint (56) guarantees that every job is processed exactly once.

Remark 7. In this model we used the following assumption: as soon as all robots have been assigned a laser source, specifing the order of the jobs on the laser sources fully describes the solution. This is not true if collisions are taken into account. Here we cannot directly check whether two robot moves overlap if they are connected to different laser sources.

Our last model is a time space network model as often used in column generation approaches. Our computational results were not obtained by column generation, though. Therefore, it is possible that the solution times of the LP relaxations for this model can be improved. However, the problem that the integrality gap of the master problem will remain fairly large because of the makespan objective cannot be cured by column generation alone. Moreover, all constraints like resource sharing involving more than one server must enter the master problem, leading to an even larger integrality gap in the master problem. This model represents approaches that can be turned into column generation models by Dantzig-Wolfe decomposition. Note that this model needs that time is discretized, and the corresponding discretization error is not neglectable, since many small, rounded durations in tours that must be synchronized finally add up to the makespan (in our tests, rounding errors w.r.t. a step-size of 0.1 s sometimes add up to 10 % of the optimal makespan!).

Let all distances be integer, and let μ be an upper bound for the makespan. Then the time periods to be considered are $\{0,1,\dots,\mu\}$. Let $G^{s,l} = (V^{s,l}, A^{s,l})$, $G^l := (V^l, A^l)$ be directed graph with node sets

(61) $$V^{s,l} := \{(j,p,t), j \in J, p \in \{j_a, j_b\}, t = 0,\dots,\mu\} \cup \{d^s\}$$

(62) $$V^l := \{(s,j,p,t), s \in S, j \in J, p \in \{j_a, j_b\}, t = 0,\dots,\mu\} \cup \{d^l\}$$

The interpretation of node $(j,p,t) \in V^{s,l}$ is the same as above with the exception that t encodes the start time of the job. The feasible solutions of the model can be interpreted as a tour for every robot and a tour for every laser source. When Robot s starts welding at node (j,p,t), then Laser Source l also has to start service at node (j,p,t). This blocks the laser source for all other robots until the processing of the job has finished.

There are three types of arcs in $A^{s,l}$. The first are starting and go-home arcs of the form (d^s,v) and (v,d^s). The second type are processing and moving arcs leading from (j_1,p_1,t_1) to $\big(j_2,p_2,t_2,\delta^s(p_1,q_1)+\delta^s(q_1,p_2)\big)$. The last type are waiting arcs of the form $(j,p,t,j,p,t+1)$. Waiting is always done in front of the first job position of a job to be processed. The arc set A^l is very similar. The only difference is the duration of a weld-and-move arc. If the source and the target node belong to the same robot, then the duration is only $\delta^s(p,q)$, otherwise the switching time τ^l must be added.

Besides the makespan variable z, there are two type of binary flow variables $x_a^{s,l}, a \in A^{s,l}$ and $y_a^l, a \in A^l$. Collision avoidance in this model would require set packing on time slots: a huge blow up with weak LP bounds.

Problem 6 (Time-Space Network (tsn)).

$$\min z$$

subject to

(63) $$\sum_{((j,p,t),d^s)\in A^{s,l}} \big(t+\delta^s(p,q)+\delta^s(q,d^s)\big)x^{s,l}_{((j,p,t),d^s)} \le z \qquad \forall\, l \in L, s \in S$$

(64) $$\sum_{\substack{l\in L\\ s\in S\\ ((j,p,t),(j',p',t'))\in A^{s,l}\\ j\neq j_2}} x^{s,l}_{((j,p,t),(j',p',t'))} + \sum_{((j,p,t),d^s)\in A^{s,l}} x^{s,l}_{((j,p,t),d^s)} = 1 \qquad \forall\, j \in J$$

(65) $$\sum_{l\in L}\sum_{(d^s,v)\in A^{s,l}} x^{s,l}_{(d^s,v)} \le 1 \qquad \forall\, s \in S$$

(66) $$\sum_{(v,w)\in A^{s,l}} x^{s,l}_{(v,w)} - \sum_{(w,v)\in A^{s,l}} x^{s,l}_{(w,v)} = 0 \qquad \forall\, l \in L, s \in S,\ v \in V^{s,l}\setminus\{d^s\}$$

(67) $$x^{s,l}_{((j,p,t),d^s)} + \sum_{\substack{((j,p,t),(j',p',t'))\in A^{s,l}\\ j\neq j'}} x^{s,l}_{((j,p,t),(j',p',t')} - y^l_{((s,j,p,t),d^l)} - \sum_{\substack{((s,j,p,t),(s',j',p',t'))\in A^l\\ j\neq j'}} y^l_{((s,j,p,t),(s',j',p',t'))} = 0 \qquad \forall\, l \in L,\ (s,j,p,t)\in V_l$$

(68) $$\sum_{(d^l,v)\in A^l} y^l_{(d^l,v)} \le 1 \qquad \forall\, l \in L$$

(69) $$\sum_{(v,w)\in A^l} y^l_{(v,w)} - \sum_{(w,v)\in A^l} y^l_{(w,v)} = 0 \qquad \forall\, l \in L, v \in V^l \setminus \{d^l\}$$

(70) $$x^{s,l}_a \in \{0,1\} \qquad \forall\, l \in L, s \in S, a \in A^{s,l}$$

(71) $$y^l_a \in \{0,1\} \qquad \forall\, l \in L, a \in A^l$$

(72) $$z \ge 0$$

(73) $$z \in \mathbb{Z}$$

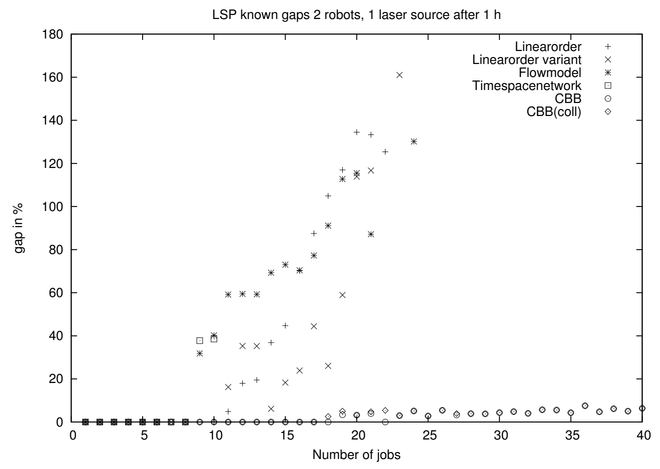


FIGURE 3. Two robots, one laser source: Known gaps after 1 h

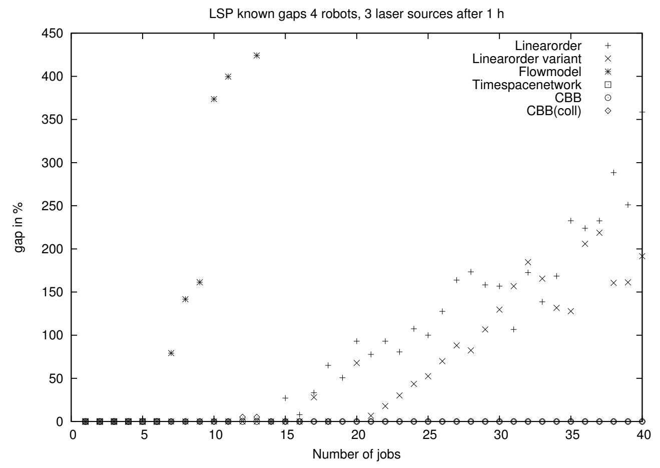


FIGURE 4. Four robots, three laser sources: Known gaps after 1 h

5.3. **Performance Metrics.** The yard stick for all test runs was a primal bound generated by our method with a time limit of one day. To this end, we ran our algorithm in two variants: CBB without collision avoidance and CBB(coll) with collision avoidance. After one day the execution was stopped, and the best found feasible solution and its cost were recorded. We call this the *yard-stick solution*, and the corresponding assignment of jobs to robots is the *yard-stick assignment*. There was not enough time to let the other models run for one day on all instances, but checking just a few instances showed CBB was the only method to generate primal bounds for every instance in one day. In the test computations we wanted to reveal why CBB performs so well on our test data.

We first compared various aspects for all methods on instances of various sizes, where the method for the MILP models of the previous subsection is to solve them by cplex with

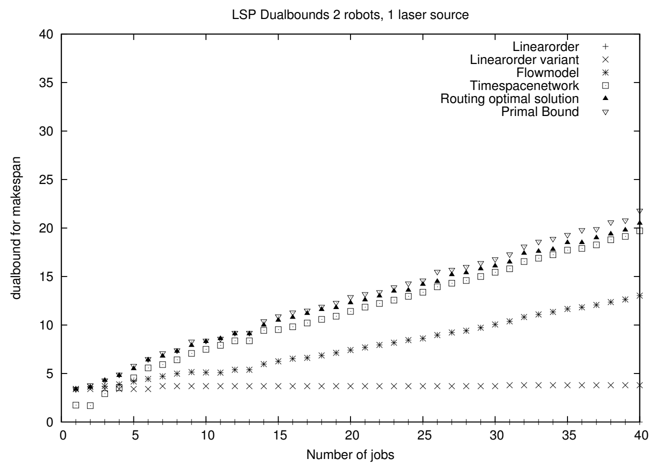


FIGURE 5. Two robots, one laser source: Value of relaxation

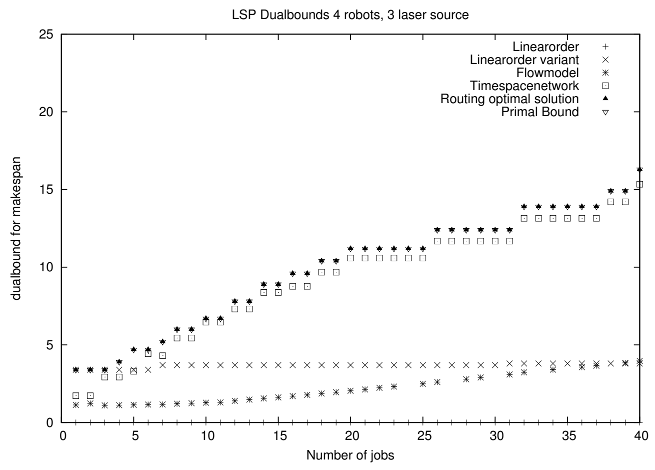


FIGURE 6. Four robots, two laser sources: Value of relaxation

default parameters (except the LP solution method, where we took the fastest one for each MILP):

- For every method, how large is the gap after one hour between the cost of its the best solution and its best dual bound? This gives a hint about how useful a method is, if it is the only one used.
- For every method, how large is the value of the corresponding relaxation (LP relaxation for MILP; combinatorial routing relaxation for CBB)? This gives a hint about how tight the model is and, therefore, how many branch-and-bound nodes cplex will need to close the gap.
- For every method, how long does it take to compute the relaxation? This gives a hint about how difficult the relaxation is and, therefore, how many branch-and-bound nodes cplex will manage to process per time unit.

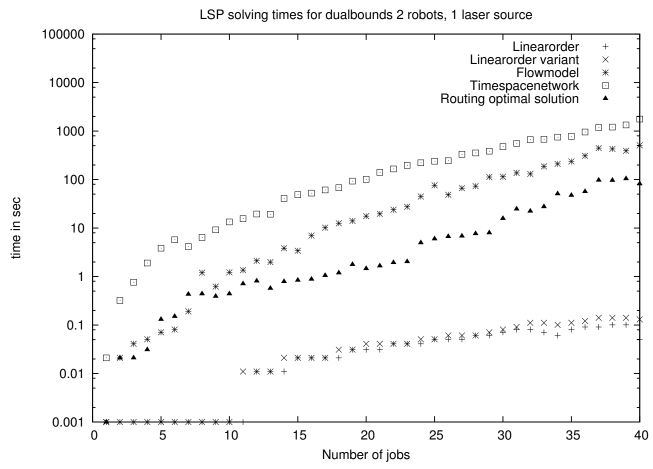


FIGURE 7. Two robots, one laser source: Solving time of relaxation

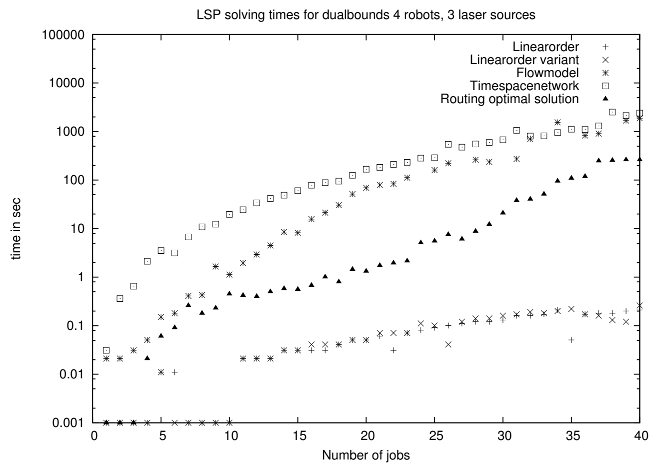


FIGURE 8. Four robots, three laser sources: Solving time of relaxation

To investigate further the outcome of these tests, we tested the methods when restricted to a fixed assignment of jobs to robots (LSP-J). It is our impression that the core of the problem lies in this subproblem. For all tests on LSP-J we fixed the variables according to the yard-stick assignments.

- For every method, how large is the value of the corresponding relaxation (LP relaxation for MILP; combinatorial LSP-s relaxation for CBB)?
- For every method, how long does it take to solve the corresponding relaxation?

All tests have been carried out on an Intel Xeon quad core processor with 2.33Ghz and 64 GB memory running ubuntu linux 8.04 in 64 bit mode. We used cplex 12.1 for all mixed integer models, including the subroutine solving RSP-T in Algorithm 1. The RSP-s

	10 jobs				20 jobs			
	lo	lov	net	tsn	lo	lov	fm	tsn
#vars	138	132	16052	105602	469	454	230022	514300
#cons	643	597	376	7937	2112	2032	1291	21059
#nz	4138	3334	92134	324965	13837	11232	1353628	1637552

	30 jobs				40 jobs			
	lo	lov	fm	tsn	lo	lov	fm	tsn
#vars	1001	978	1131992	1435406	1734	1704	3777002	3372946
#cons	4577	4459	2806	39839	8338	8178	4981	68835
#nz	30351	24864	6708656	4614926	55768	45694	22458970	10940231

TABLE 1. Selected problem sizes for LSP, two robots, one laser source

	10 jobs				20 jobs			
	lo	lov	fm	tsn	lo	lov	fm	tsn
#vars	148	142	26552	231122	479	464	421022	1206350
#cons	1077	1031	1106	19509	3510	3430	3831	55501
#nz	7764	7280	151122	676786	25487	23826	2473548	3703963
	30 jobs				40 jobs			
	lo	lov	fm	tsn	lo	lov	fm	tsn
#vars	1011	988	1903052	2835248	1744	1714	6051962	6610220
#cons	7831	7713	8356	90179	14356	14196	14861	156951
#nz	57521	54322	11267976	8719915	14356	100600	35966622	20497264

TABLE 2. Selected problem sizes for LSP with four robots, three laser sources

subproblems were solved by our own branch-and-cut solver implemented in the framework SCIP 1.2. Here, cplex was used as the underlying LP solver.

5.4. **Results.** The first test ran all mixed integer models as well as our algorithms CBB and CBB(coll) on all instances with time limit one hour. Figure 3 shows the known gap for one laser source, two robots. Figure 4 shows the same for four robots and three laser sources.

Instances with two robots and up to eight jobs could be solved by any algorithm to optimality, while in none of the MILP models cplex could obtain a primal feasible solution for more then 25 jobs. With four robots, it seems to be easier for cplex to obtain a primal bound for the MILP models, but the gaps are still huge for more than ten jobs. This is getting worse when more laser sources are available, probably because the assignment of robots to laser sources is now part of the optimization process. Our combinatorial branch-and-bound algorithms CBB was able to solve all four-robots instances with three laser sources to optimality and always found a primal solution with a small gap for the one-laser source instances; this performance did not suffer, when collision avoidance was observed by CBB(coll).

Of course, the time limit of one hour is arbitrary, and in a close call between various methods corresponding results might be misleading. However, we tried to find time limits more in favor of any of the other methods: we did not find any.

More systematic findings are revealed by the structural results on relaxation effectivity and efficiency. Figures 5 and 6 show the dual bounds coming from the LP relaxation of the instances. The time needed to solve these is given in 7 and 8. It is apparent that both LP relaxations of the linear ordering models are fast to solve but not tight enough, whereas the LP relaxation of the time-space network model is tighter but takes way too much time to solve for a successful use in a branch-and-bound scheme. The flow model is somewhere in the middle, but degrades in performance when four robots need to share three laser sources.

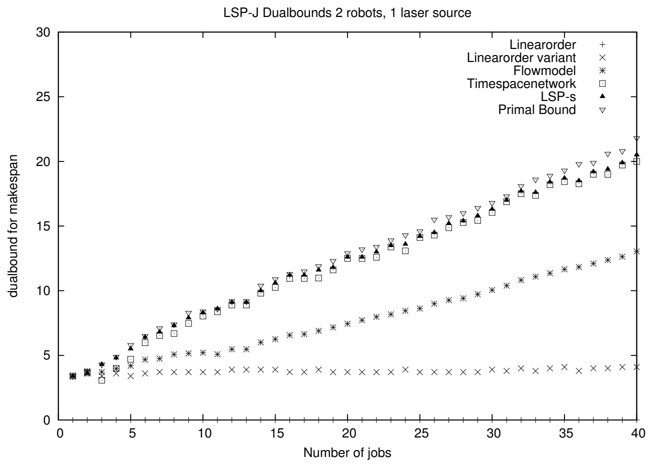


FIGURE 9. Two robots, one laser source: Value of relaxation

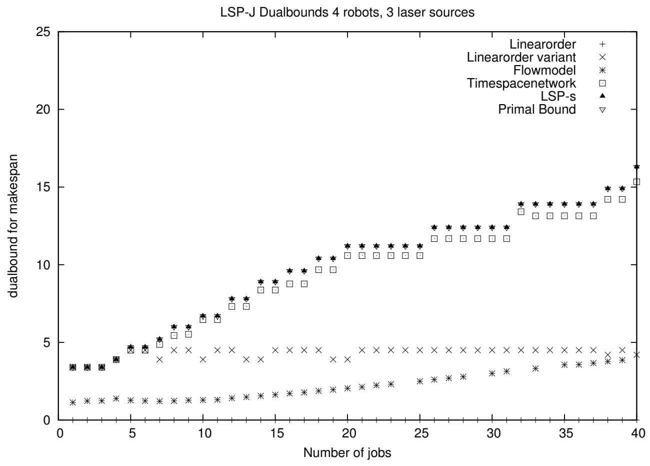


FIGURE 10. Four robots, two laser sources: Value of relaxation

The solution times for the LP relaxations can be understood by looking on the problem sizes: The sizes of the mixed integer models of selected instances can be found in Tables 5.3 and 5.3. The time-space network model becomes very large when the number of jobs is increasing. The larger instances could not be solved by the primal or dual simplex method; the barrier had to be used instead.

So, why do CBB and CBB(coll) work better than cplex on the MILP-models? The main reason lies in the approach to the subproblem LSP-J and dual bounds for it. Recall that CBB receives the relevant lower bounds from the solution of instances of LSP-s, an NP-hard single-server routing problem. The MILP methods use LP relaxations as dual bounds.

We therefore performed tests on LSP-J. The data is generated by prescribing the yardstick assignment – the best solution found by CBB or CBB(coll) after one day. By this fixing, the size of the linear order based models changes only marginally, so we just specify

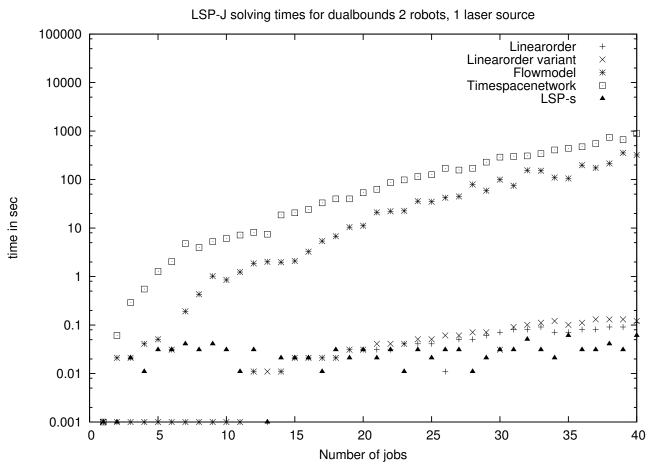


FIGURE 11. Two robots, one laser source: Solving time of relaxation

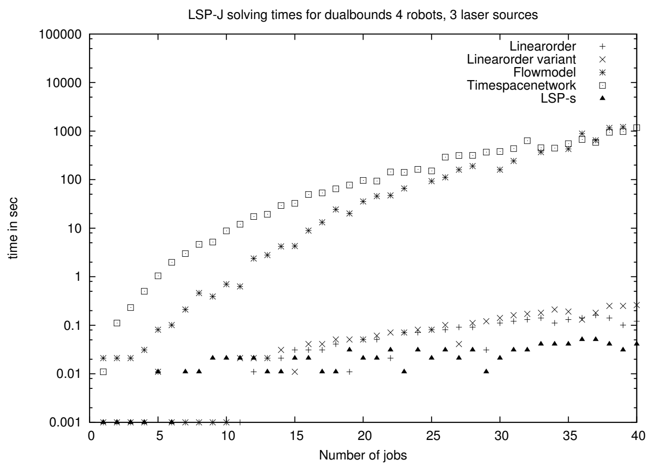


FIGURE 12. Four robots, three laser sources: Solving time of relaxation

the size of the flow model and the time-space network, see Tables 3 and 5. In order to indicate the problem size of the LSP-s relaxation, we specify for it the number of nodes in each robot routing graph; this is twice the number of jobs plus one for the depot position, see Tables 4 and 6.

Figures 9 and 10 give the dual bound of the root relaxation for LSP-J. The computation times are given in Figures 11 and 12. As we can see, the LSP-s relaxation provides the strongest bound, and the computation times show that the solution of the LSP-s relaxation scales very well with respect to the number of jobs. The time-space network approach suffers from extremely long computation times, and using it as an often-called subproblem in a solution process seems prohibitive. This even more so, since its bounds are still worse than the ones from LSP-s, which can be obtained much faster by our polyhedral ATSP solver. The aggregated picture of results in Figure 13 shows very well that the LSP-s bound is

	10 jobs		20 jobs		30 jobs		40 jobs	
	fm	tsn	fm	tsn	fm	tsn	fm	tsn
#vars	7612	55670	139942	331316	734432	952750	2383882	2172360
#cons	296	5723	1091	16935	2386	32425	4181	55225
#nz	43478	1069312	822926	1049625	4351438	3058073	14173046	7034732

TABLE 3. Selected problem sizes for LSP-J with two robots, one lasersource

	10 jobs	20 jobs	30 jobs	40 jobs
r_1	11	19	31	43
r_2	11	23	31	39

TABLE 4. Number of nodes in LSP-s relaxation for two robots

	10 jobs		20 jobs		30 jobs		40 jobs	
	fm	tsn	fm	tsn	fm	tsn	fm	tsn
#vars	12032	122474	278462	793358	1159292	1865378	3856922	4273994
#cons	866	14049	3231	44617	7096	733331	12461	125907
#nz	67794	354220	1634754	2436589	6859914	5708605	22915554	13241668

TABLE 5. Selected problem sizes for LSP-J with four robots, 3 laser source

	10 jobs	20 jobs	30 jobs	40 jobs
r_1	3	3	11	15
r_2	9	19	21	27
r_3	5	11	19	27
r_4	7	11	13	15

TABLE 6. Number of nodes in LSP-s relaxation for four robots

always inside a rectangle of good effectivity and good efficiency, whereas the other methods either are ineffective for many instances or inefficient. Note, that the scale is logarithmic.

One might conjecture that the good quality of the routing-based bound of the LSP-s enables us to hierarchically decompose the problem by a route-first-schedule-second approach and still achieve good solutions. Figure 14 shows the gaps of solutions found by taking routing-optimal solutions (no resource sharing at all) and laser-sharing optimal solutions (no collision avoidance), resp., and performing a collision-aware rescheduling on them. In more than neglectably many of our instances this leads to substantial gaps. For example, if the possible makespan with one instead of two laser sources is 30 s, a gap of 10 % means that only a solution with makespan 33 s can be found. If the process cycle time is, say, 32 s, this is the difference between halving the investments for laser sources, i.e., tremendous savings, and nothing.

One more thing: Collision avoidance might appear nit-picking at first sight because in many cases the *makespan* does not increase too much. In order to illustrate how different the corresponding *solutions* can be, we show in Figure 15 a snapshot of an optimal solution without collision avoidance and two snapshots of an optimal solution with collision avoidance. It is pretty clear that the solution with collision avoidance has a totally different routing. This routing leads to a solution where the robots are never in the middle area at the same time, whereas the solution without collision avoidance allows both robots to meet in the middle. A route-first-schedule-second approach would just let one of the robots wait so that collisions are avoided; yet, their routings would still let them come close in the middle.

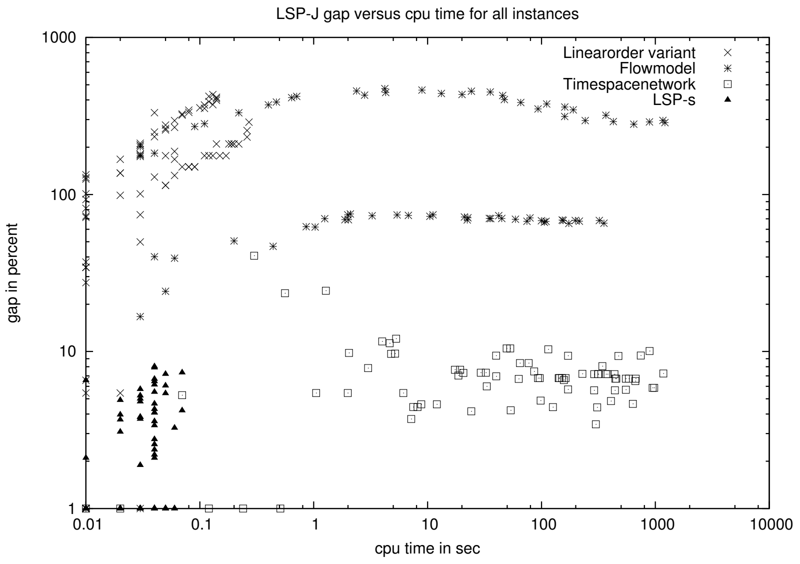


FIGURE 13. Solution times and gaps of all instances

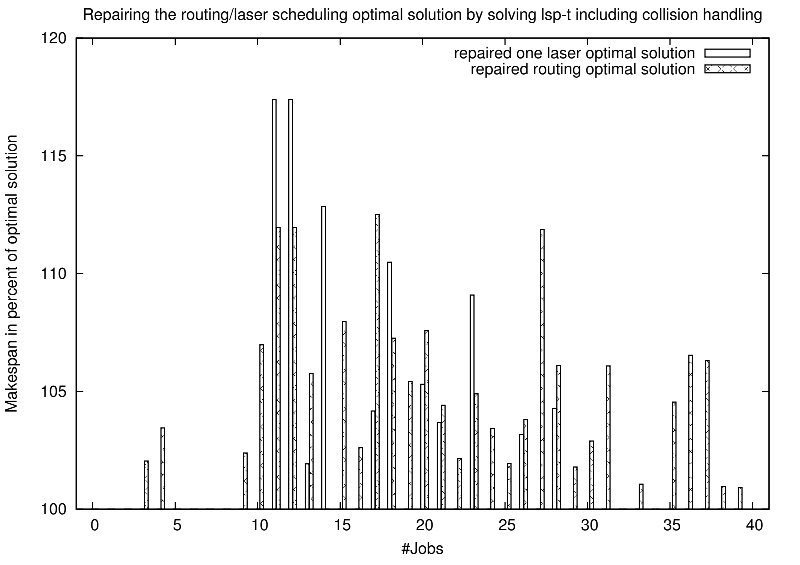


FIGURE 14. Gaps for route-first-schedule-second

It is very likely that the more conservative solution produced by our concept of collision avoidance would be favored by most engineers.

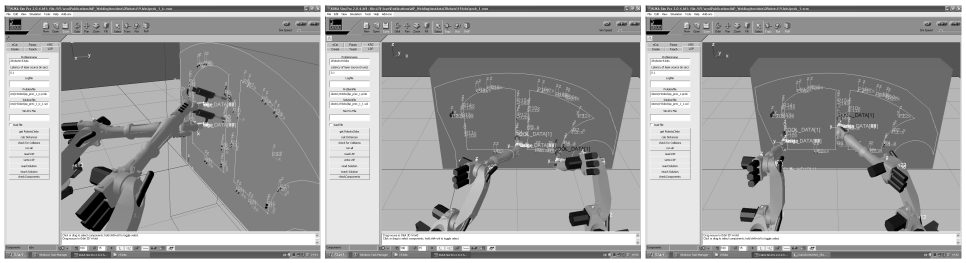


FIGURE 15. Snapshots of an optimal collision-ignoring (left) and an optimal collision-avoiding (middle and right) solution

6. CONCLUSIONS

We have introduced the laser sharing problem in the strategic planning of the welding process in car body manufacturing. The application poses three structural problems that make it particularly difficult and interesting: it combines routing and scheduling, the makespan has to be minimized, and proven optimality is needed to decide the question about how many laser sources are needed to process all welding tasks in the process cycle time.

The laser sharing problem fits into a general framework that combines routing and scheduling aspects at the same time. As a generic model, we presented the Routing & Scheduling Problem (RSP), for which we designed and analyzed a combinatorial branch-and-bound algorithm. In particular, we presented a pseudo-polynomial time dynamic-programming algorithm for the important scheduling subproblem with given routings when the number of resources is bounded.

The global algorithm heavily utilizes exact solutions to NP-hard combinatorial subproblems for dual bounds. We showed that on instances with industry-type characteristics of the laser sharing problem we started with, this algorithm is the first to solve instances of industrial scale close to optimality with certificate.

We think that the combination of routing and scheduling via resource sharing plays a role in many different contexts. The RSP provides a unified framework to investigate examples and to formulate algorithms. It is most likely that the performance of algorithms will depend on the characteristics of data in the respective application domain. Therefore, a lot has to be done to clarify the global landscape of routing and scheduling.

REFERENCES

[1] D.L. Applegate, R.E. Bixby, V. Chvátal, and W.J. Cook. *The Traveling Salesman Problem – A Computational Study*. Princeton University Press, 2006.

[2] Enrique Benavent, Alessandro Carrotta, Angel Corberán, José M. Sanchis, and Daniele Vigo. Lower bounds and heuristics for the windy rural postman problem. *Eur. J. Oper. Res.*, 176(2):855–869, 2007.

[3] Enrique Benavent, Angel Corberán, Estefanía Piñana, Isaac Plana, and José M. Sanchis. New heuristic algorithms for the windy rural postman problem. *Comput. Oper. Res.*, 32(12):3111–3128, 2005.

[4] P. Brucker, A. Drechsel, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operations Research*, 112:3–41, 1999.

[5] P. Brucker and S. Knust. *Complex Scheduling*. Springer-Verlag, Berlin, Heidelberg, New York, 2006.

[6] E. Demeulemeester. *Optimal Algorithms for various classes of multiple resource-CPCPs*. PhD thesis, Katholik Universiteit Leuven, 1996.

[7] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constraint routing and scheduling. In Michael Ball, Tom Magnanti, Clyde Monma, and George Newhauser, editors, *Network Routing*, volume 8 of

Handbooks in Operations Research and Management Science, chapter 2, pages 35–140. Elsevier, Amsterdam, 1995.

[8] Bertsekas Dimitri P.˙*Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 2nd edition, 2000.

[9] Matteo Fischetti, Andrea Lodi, and Paolo Toth. Exact methods for the asymmetric traveling salesman problem. Gutin, Gregory (ed.) et al., The traveling salesman problem and its variations. Dordrecht: Kluwer Academic Publishers. Comb. Optim. 12, 169-205 (2002)., 2002.

[10] M.R. Garey and D.S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[11] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H Korte, editors, *discrete optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. North-Holland Publishing Company, Amsterdam, New York, Oxford, 1979.

[12] M. Grötschel, H. Hinrichs, K. Schröer, and A. Tuchscherer. Ein gemischt-ganzzahliges lineares Optimierungsproblem für ein Laserschweißproblem im Karosseriebau. *Zeitschrift für wissenschaftlichen Fabrikbetrieb*, 5:260–264, 2006.

[13] C. Hempsch and S. Irnich. Vehicle-routing problems with inter-tour resource constraints. In B.L. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, 2007.

[14] S.O. Krumke, J. Rambau, and L.M. Torres. Realtime dispatching of guided and unguided automobile service units with soft time windows. In R. Möhring et al., editor, *Algorithms – ESA 2002*, volume 2461 of *LNCS*, pages 637–648. Springer, 2002.

[15] A. Munier and F. Sourd. Scheduling chains on a single machine with non-negative time lags. *Math. Methods Oper. Res.*, 57(1):111–123, 2003.

[16] Michael L. Pinedo. *Scheduling. Theory, algorithms, and systems. With CD-ROM. 3rd ed.* New York, NY: Springer. xvii, 671 p., 2008.

[17] J. Rambau and C. Schwarz. On the benefits of using NP-hard problems in branch & bound. In *Operations Research Proceedings 2008*, pages 463–468. Springer, 2009.

[18] J. Rambau and C. Schwarz. How to avoid collisions in scheduling industrial robots? Preprint, Universität Bayreuth, 2010.

[19] T. Schneider. Ressourcenbeschränktes Projektscheduling zur optimierten Auslastung von Laserquellen im Automobilkarosseriebau. Diplomarbeit, University of Bayreuth, 2006.

[20] C. Schwarz and J. Rambau. Optimal dispatching of welding robots. Slides for the 13th Combinatorial Optimization Workshop, January 2009.

[21] Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

[22] W. Welz. Route planning for robot systems. Diplomarbeit, Technische Universität Berlin, 2010.

[23] W. Welz and M. Skutella. Route planning for robot systems. Talk WD-07.1 of the International Conference Operations Research, 2010.

[24] E.D. Wikum. *One-Machine Generalized Precedence Constrained Scheduling*. PhD thesis, School of Industrial and Systems Enigeering, Georgia Institute of Technology, 1992.

[25] E.D. Wikum, D.C. Llewellyn, and G.L. Nemhauser. One-machine generalized precedence constrained scheduling problems. *Operations Research Letters*, 16:87–99, 1994.

J. RAMBAU AND C. SCHWARZ LS WIRTSCHAFTSMATHEMATIK, UNIVERSITÄT BAYREUTH, GERMANY, TEL.: +49-921-55-7350, FAX: +49-921-55-7352,

E-mail address: {firstname.lastname}@uni-bayreuth.de