SOLVING A VEHICLE ROUTING PROBLEM WITH RESOURCE CONFLICTS AND MAKESPAN OBJECTIVE WITH AN APPLICATION IN CAR BODY MANUFACTURING

JÖRG RAMBAU AND CORNELIUS SCHWARZ

ABSTRACT. We investigate the problem of dispatching arc welding robots in car body manufacturing. Such arc welding robots receive their energy from expensive laser sources. Laser sources can be shared among the robots. However, this requires that the robots are scheduled because each laser source can only be used by one robot at a time. We want to compute the minimal number of laser sources necessary to perform all welding tasks in a given processing time. To this end, we introduce the Laser Sharing Problem (LSP): for a given number of laser sources, find collision-free scheduled tours for all robots through all welding jobs so that the makespan is minimized. We propose a branch-and-bound algorithm for the LSP using bounds that stem from optimal solutions to carefully selected NP-hard combinatorial subproblems. This is the first algorithm for the LSP that is able to solve industrially relevant problem scales.

1. INTRODUCTION

The investigations in this paper were directly motivated by a real-world problem in car body manufacturing [5]: 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 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. We directy proceed to the set-up of a welding cell: 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 suffices to specify the assignment and order of jobs for each robot. However, in practice this leads to 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 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. That is, 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. 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 (*LSP*)* (collision avoidance is ignored for the moment).

The resource constraints and the makespan objective make the LSP hard and interesting at the same time. In this paper, we show how to solve industry-scale instances of it with performance guarantees. To this end, we propose a branch-and-bound method, which is based on bounds coming from the solutions of NP-hard subproblems. These are solved to optimality by means of state-of-the-art MILP techniques. The investigation of the relations between various subproblems of the laser sharing problem plays a key role in the design of the dual bounds.

Why do we deliberately choose to solve NP-hard subproblems? Branch-and-bound is 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 large-scale instance of LSP requires only small-scale instances for the NP-hard subproblems – given state-of-the art MILP 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 among 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 [15, 16].

Short, preliminary versions of the algorithm for the laser sharing problem without collision avoidance have already been presented together with preliminary computational results in [11]. The incorporation of collision avoidance required the extension of the Tuchscherer model from [5]. 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 [13] and formally introduced in [12].

1.1. Related Work.

In the vehicle routing literature, the term *scheduling* usually refers to capacity or time window constraints [14, 3]. These have been generalized by *resource extended functions (REFS)* [8]. Using REFS one can measure the resource consumption along a vehicle paths. This is especially useful in resource constraint shortest path problems. Hempsch and Irnich [6] also extended this concept to *global resources*, i.e., resources that affect more than one vehicle. However, these are *non renewable* while for laser source sharing and collision avoidance we need *renewable resources*. For this REFS cannot be used.

Recently Welz and Skutella [16] studied point welding robots. They purpose was to find collision free robot tours. For this they proposed a column generation based approach where they integrated the cycle time as a hard constraint and the objective is to minimize operating costs instead of the makespan. The important difference to arc welding robots is the absence of laser source sharing. Determining the minimum number of required laser sources makes an infeasibility proof mandatory in LSP.

In her PhD thesis, Knust [10] analyzed shop scheduling problems with transportation robots. In this situation the robots are used to transfer the jobs between the machines. A transportation move can be modeled as an additional task to be processed on some robot machine. After assigning robots to the transportation operations we obtain a shop scheduling problem where the robot moves are modeled as disjunction which are weighted with the robot driving times.

None of the exact approaches known to us can handle sharing of renewable resources and a makespan objective at the same time.

1.2. **Our contribution.**

In this paper, our comprehensive study of the laser sharing problem goes beyond the conference presentations in [11, 12] in the following aspects: We present

- a unified framework for shared resources which generalized laser sources as well as collision avoidance,
- a report on comprehensive computational results for our algorithm applied to the laser sharing problem with laser sharing and collision avoidance, where the data is based on an industry-standard simulation tool and a geometrically plausible (yet artificial) set of welding jobs.

The result most relevant for practice can be summarized as follows: our LSP-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.

1.3. **Overview of this paper.**

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. Next, in Section 3 we describe the ideas, the building blocks, and the overall design of the new algorithm CBB. In Section 4.3 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 5.

2. PROBLEM STATEMENT

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. For every Server s let $G^s = (V^s, A^s)$ be the complete directed graph with node set $V^s := \{d^s\} \cup \{j_a, j_b \mid j \in J^s\}$. *Processing* a job means to traverse either arc (j_a, j_b) or (j_b, j_a).

For $(p,q) \in A^s$ let $\delta^s(p,q) \geq 0$ denote the time that Robot s needs to get from p to q. If p and q are end positions of the same welding job, then the move (p,q) is done in welding mode, which means that Robot s is 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 $\tau^l \geq 0$.

A *line-line collision* is a pair $(a,a') \in C_{\text{ll}}$ of arcs $a \in A^s, a' \in A^{s'}$ with $s, s' \in S$. The meaning of this data is that the moves of s along a and of s' along a' must not overlap in time. The motivation is as follows: Whenever there are positions q^s on a and $q^{s'}$ on a' for which the geometries of the robots s and s' are not disjoint, a collision cannot be excluded. The restriction to seperate such movements completely in time is the most conservative modelling decision.

Similarly, a *line-point collision* $(a, v') \in C_{\text{lp}}$ means: Robot s' residing at Node v' will collide with Robot s moving along Arc a. Here, we do not allow Robot s' to visit v' while s is processing a. Note that v' is also an intermediate point for every arc $(v, v'), (v', v) \in A^{s'}$. Thus line-point collisions always induces line-line collisions.

In the following, bars on symbols indicate decisions. The task in the LSP is to compute a *scheduled dispatch* $\bar{D} := (\bar{T}^s, \bar{t}^s, \bar{r}^s)_{s \in S}$, i.e., to assign to each robot $s \in S$ a set of jobs $\bar{J}^s$ to process, a tour $\bar{T}^s = (d^s = v_1, \ldots, v_{n^s} = d^s)$ in G^s, a laser source $\bar{r}^s \in L$ used for welding, and a schedule $\bar{t}^s$ that assigns a departure time $\bar{t}^s(v)$ to each visited node v in the tour $\bar{T}^s$ such that

- each job j is processed in exactly one tour $\bar{T}^s$;
- jobs assigned to robots s, s' sharing a laser source $\bar{r}^s = \bar{r}^{s'}$ do not overlap in time with respect to the corresponding schedulings $\bar{t}^s$ and $\bar{t}^{s'}$;
- 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.

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.

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.

We will now propose a general model for shared resources. It covers the laser source sharing as well as collision avoidance, which can be seen as some kind of sharing a space resource.

In vehicle routing, resources traditionally mean capacity restrictions or time windows [14]. These have been generalized by resource extended functions [9]. However, in the LSP we are faced with *renewable* resources. Such resources are common, e.g., in project scheduling (see for instance [2]). Therefore, our approach encompasses completely new situations that have been investigated neither in the vehicle routing nor in the scheduling literature so far.

The key idea for a unified model for laser source sharing and collision avoidance is to distinguish between *resource types* and *resources*. A resource type R is a set of resources $R = \{r_1, \dots, r_h\}$. For example, *laser source* is a resource type, consisting of all the individual laser sources available to the robots. The individual laser sources are resources.

To every arc a and every node v, there is a set $\rho(a)$ and $\rho(v)$, resp., of required resource types. Before a robot can traverse a or reside at v, a specific resource has to be selected from each required resource type. This will be called *resource selection*. All selections are applied *globally*, i.e. two arcs or nodes of the robot's tour wich require the same resource type will be processed with same resource, e.g. switching a laser source along a path is not allowed.

An ordinary resource such as a line-line collision can be modeled as a resource type containing only a single resource. In such cases the resource selection is trivial.

The time for a resource $r \in R$ to switch to a different robot will be denoted by τ^r. For the resource "laser source", this is the delay to switch to another robot, for the collision resources this can be used to seperate movements with potential collisions in an even more conservative fashion.

3. A Combinatorial Branch and Bound Algorithm

Our algorithm is a two-phase branch-and-bound approach: The first phase will reduce LSP to a usually small set of LSP-J instances. In the second phase, each of these instances is solved to optimality. Throughout the algorithm we will make use of subproblems of type LSP-s and of type LSP-T. The main observation is that, although both subproblems are NP-hard, they can be solved to optimality very fast for the relevant problem scales. This provides us with strong lower bounds, which are the key in solving industrial scale instances of LSP to optimality. In the section on computational results, we will compare these bounds in both evaluation time and quality to an integer programming model for LSP based on a time-space network. These models are known to provide good lower bounds, and some readers might suspect that a model of that type should be able to solve the LSP, too.

We start with a mixed integer programming formulation that we use to solve the LSP-s and LSP-T subproblems. Then we describe the phase-two algorithm for LSP-J. In the last subsection we cover the phase-one method and the top level algorithm.

We will use $\ell(\bar{T}^s)$ to denote the length of a tour, i.e., the sum over the distances of all used arcs. For 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. This is the time when the last server is back to its home position after all requests have been served.

3.1. **The Single Server Problem: LSP-s.**

The single server problem LSP-s is a pure routing problem on the complete directed graph $G = (V, A)$ with vertex set $V := \{d^s\} \cup \{j_a, j_b \mid j \in \bar{J}(s)\}$. It can be modelled as an *asymmetric traveling salesman problem* with the additional constraint that for every job $j \in \bar{J}(s)$ one of the two arcs (j_a, j_b) or (j_b, j_a) must be part of the solution. We use the following integer programming formulation:

Problem 1 (LSP-s).

$$\min \sum_{(v,w) \in A} \delta^s(v,w) x_{v,w}$$

subject to

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

$$\sum_{(v,w)\in A} x_{v,w} = 1 \qquad \forall\, w \in V \tag{2}$$

$$\sum_{(v,w)\in A\cap(U\times U)} x_{v,w} \leq |U|-1 \qquad U \subset V, 2 \leq |U| \leq |V|-2 \tag{3}$$

$$x_{j_a,j_b} + x_{j_b,j_a} = 1 \qquad \forall\, j \in \bar{J}(s) \tag{4}$$

$$x_{v,w} \in \{0,1\} \qquad \forall\, (v,w) \in A \tag{5}$$

Note that this is just the standard ATSP model (see for instance [4]) with additional constraints (4). These additional constraints guarantee that all jobs are processed. Since the convex hull of all LSP-s-solutions is contained in the ATSP polytope, every valid inequality from the ATSP literature can be used as a cutting plane for the LSP-s as well. We denote by ${T_{\text{LSP-s}}}^s(J,v)$ an optimal tour for the LSP-s for Server s starting at some node v, processing all jobs in J and ending at d^s.

3.2. **The Fixed Dispatch problem: LSP-T.** In the LSP-T every robot has a prescribed tour $\bar{T}^s = (v_1,\dots,v_{n^s})$. Let $\rho^s(\bar{T}^s)$ denote the set of all required resource types by arcs and nodes of the tour. The goal is to find a resource selection $\bar{r}^s$ and a makespan minimal schedule $(\bar{t}^s)_{s\in S}$ without resource conflicts.

Our mixed integer programming model is based on a model that was proposed by Grötschel, Hinrichs, Schroer, and Tuchscherer [5] for the LSP-T without collision avoidance. 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

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

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. This will be modeled using binary linear ordering variables y_{v_i,w_j} where v_i, w_j are the tails of the arcs. Here, $y_{v_i,w_j} = 1$ means, that arc (v_i, v_{i+1}) is processed by Server s before s' is allowed to use (w_i, w_{i+1}). With M sufficiently large we demand for all resource types R shared by these arcs and all $r \in R$

$$x_i^s + \delta^s(v_i, v_{i+1}) + \tau^r - x_j^{s'} \leq M(3 - u_{s,r} - u_{s',r} - y_{v_i,w_j})\,.$$

Because a server blocks all local resources while residing at a node, node resources must be handled differently. Recall that the resource demand of v_i is also part of (v_i, v_{i+1}) and (v_{i-1}, v_i). Thus, if the arc (w_j, w_{j+1}) shares a resource with v_i we only traverse this arc before (v_{i-1}, v_i) is started or after (v_i, v_{i+1}) has been completed:

$$y_{v_{i-1},w_j} = y_{v_i,w_j} \quad \text{whenever } v_i \text{ and } (w_j, w_{j+1}) \text{ are in conflict}\,.$$

This can be written as

$$y_{v_{i-1},w_j} - y_{v_i,w_j} \le 2 - u_{s,r} - u_{s',r}$$
$$y_{v_i,w_j} - y_{v_{i-1},w_j} \le u_{s,r} + u_{s',r} - 2\,.$$

If a conflict for a node resource occurs at a depot, i.e., $i = 1$ or $i = n$, then the move (w_j, w_{j+1}) has to happen after the move (v_1, v_2) and before the move (v_{n-1}, v_n):

$$1 - y_{v_1,w_j} \le 2 - u_{s,r} - u_{s',r}$$
$$1 - y_{w_j,v_{n-1}} \le 2 - u_{s,r} - u_{s',r}\,.$$

We now summarize the whole model:

Problem 2 (LSP-T).

$$\min z \tag{6}$$

subject to

$$x^s_{n^s} \le z \qquad \forall\, s \in S \tag{7}$$

$$x^s_i + \delta^s(v_i, v_{i+1}) \le x^s_{i+1} \qquad \forall\, s \in S, i = 1, \dots, n^s - 1 \tag{8}$$

$$\begin{aligned} x^s_i + \delta^s(v_i, v_{i+1}) \\ + \tau^r - x^{s'}_j \le M(3 - u_{s,r} \\ - u_{s',r} - y_{v_i,w_j}) \end{aligned} \qquad \begin{aligned} &\forall\, s, s' \in S, s \neq s', \\ &i = 1, \dots, n^s - 1, \\ &j = 1, \dots, n^{s'} - 1, \\ &R \in \rho^s(v_i, v_{i+1}) \cap \rho^{s'}(w_j, w_{j+1}), \\ &r \in R \end{aligned} \tag{9}$$

$$y_{v_{i-1},w_j} - y_{v_i,w_j} \le 2 - u_{s,r} - u_{s',r} \qquad \begin{aligned} &\forall\, s, s' \in S, s \neq s' \\ &i = 2, \dots, n^s - 1, \\ &j = 1, \dots, n^{s'} - 1 \\ &R \in \rho^s(v_i) \cap \rho^{s'}(w_j, w_{j+1}), \\ &r \in R \end{aligned} \tag{10}$$

$$y_{v_i,w_j} - y_{v_{i-1},w_j} \le 2 - u_{s,r} - u_{s',r} \qquad \begin{aligned} &\forall\, s, s' \in S, s \neq s' \\ &i = 2, \dots, n^s - 1, \\ &j = 1, \dots, n^{s'} - 1 \\ &R \in \rho^s(v_i) \cap \rho^{s'}(w_j, w_{j+1}), \\ &r \in R \end{aligned} \tag{11}$$

$$1 - y_{v_i,w_j} \le 2 - u_{s,r} - u_{s',r} \qquad \begin{aligned} &\forall\, s, s' \in S, s \neq s' \\ &i = 1, j = 1, \dots, n^s - 1 \\ &R \in \rho^s(d^s) \cap \rho^{s'}(w_j, w_{j+1}), \\ &r \in R \end{aligned} \tag{12}$$

$$1 - y_{v_j,w_i} \le 2 - u_{s,r} - u_{s',r} \qquad \forall\, s, s' \in S, s \neq s' \tag{13}$$

$$\begin{aligned}
& && i = n^s - 1, j = 1, \dots, n^{s'} - 1 \\
& && R \in \rho^s(d^s) \cap \rho^{s'}(w_j, w_{j+1}), \\
& && r \in R \\
&(14) \quad & y_{v,w} + y_{w,v} = 1 \quad & \forall\, s, s' \in S, s \neq s', \\
& && v \in V^s(\bar{T}^s), w \in V^{s'}(\bar{T}^{s'}) \\
&(15) \quad & \sum_{r \in R} u_{s,r} = 1 \quad & \forall\, s \in S, R \in \rho^s(\bar{T}^s) \\
&(16) \quad & x_i^s \geq 0 \quad & \forall\, s \in S, i = 1, \dots, n^s \\
&(17) \quad & z \geq 0 \quad & \\
&(18) \quad & y_{v,w} \in \{0,1\} \quad & \forall\, s, s' \in S, s \neq s', \\
& && v \in V^s(\bar{T}^s), w \in V^{s'}(\bar{T}^{s'}) \\
&(19) \quad & u_{s,r} \in \{0,1\} \quad & \forall\, s \in S, R \in \rho^s(\bar{T}^s), r \in R
\end{aligned}$$

3.3. **Phase 2: Solving the LSP-J.** Let $\bar{J} : S \to 2^J$ be a given assignment of jobs to servers. This is equivalent to an ordinary LSP with $J^s = \bar{J}(s)$, i.e., for each server only the jobs assigned to it are feasible for it. Our branch-and-bound approach now works as follows: Assume we are given a partial tour $\bar{T}^s = (v_1, \dots, v_{i_s})$ for each server. The sets of jobs visited by Server s in $\bar{T}^s$ 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 $T_{\text{LSP-s}}{}^s(K^s, v_{i_s})$. Hence,

$$\max_{s \in S} \Big(\ell(\bar{T}^s) + \ell\big(T_{\text{LSP-s}}{}^s(K^s, v_{i_s})\big) \Big) \tag{20}$$

is a valid lower bound. In order to further improve this bound, we make use of the resource demands of the partial tours. For this we use the following fact: Every tour for s starting with $\bar{T}^s$ needs at least $\ell(T_{\text{LSP-s}}{}^s(K^s, v_{i_s}))$ time units to move from v_{i_s} back to d^s. Denote by $T_{\text{LSP-T}}(\bar{T}, l)$ an optimal scheduled dispatch of the LSP-T problem for the tours $\bar{T}$ but where the distance from the last position to the home position $\delta^s(v_{n^s}, d^s)$ is replaced by l^s. The effect of this modification is that the new go-home time in LSP-T is now the minimal time the server needs to serve the requests not handled in LSP-T and to go back to its home position. As a consequence, no feasible scheduled dispatch starting with the given partial tours can finish earlier than the makespan of the resulting scheduled dispatch.

Lemma 1. *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 $l := (l^s)_{s \in S}$, $l^s := \ell(T_{LSP\text{-}s}{}^s(K^s, v_{i_s}))$. Then we have*

$$\max_{s \in S} \big(\ell(\bar{T}^s) + l^s \big) \leq \ell\big(T_{LSP\text{-}T}(\bar{T}, l)\big) \leq \ell(\bar{D}). \quad \square$$

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 vertex $v \in \{j_a, j_b\}$. A leaf consists of a dispatch, i.e., a full tour for each server. Finally, we evaluate optimal schedules in the leaves by solving the corresponding LSP-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 1. Algorithm 1 summarizes the method. Note, if $\bar{J}$ corresponds to the job-server assignment of an optimal solution to the LSP, then Algorithm 1 will find an optimal solution of the LSP.

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

```
Require: Data of the LSP-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 LSP-J.
  $\bar{T}^s(N) \leftarrow (d^s), s \in S$
  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
      $K^s \leftarrow J^s \setminus J(\bar{T}^s(N))$
      $v^s \leftarrow \text{endpos}(\bar{T}^s(N))$
      evaluate $T_{\text{LSP-s}}{}^s(K^s, v^s)$ and set $l^s$ to its length
    end for
    evaluate $T_{\text{LSP-T}}(\bar{T}(N), l)$ for $l = (l^s)_{s\in S}$
    $\lambda(N) \leftarrow \ell\big(T_{\text{LSP-T}}(\bar{T}(N), l)\big)$
    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 $\bar{T}^s(N)$
          $\bar{T}^s_{\text{OPT}} \leftarrow \bar{T}^s(N)$
          set $\bar{r}^s_{\text{OPT}}, \bar{t}^s_{\text{OPT}}$ according to $T_{\text{LSP-T}}(\bar{T}(N), l)$
        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 $\bar{T}^{\hat{s}}(N)$
          create node $N_2$ with $(j_b, j_a)$ added to $\bar{T}^{\hat{s}}(N)$
          add $N_1$ and $N_2$ to set of nodes
        end for
      end if
    end if
  end while
  return $(\bar{T}^s_{\text{OPT}}, \bar{t}^s_{\text{OPT}}, \bar{r}^s_{\text{OPT}})_{s\in S}$
```

Remark 2. Let $\bar{T}^s_{\text{concat}}$ be the concatenation of $\bar{T}^s$ and $T_{\text{LSP-s}}{}^s(K^s, v_{i_s})$. Solving the LSP-T with a tour set $\bar{T}_{\text{concat}} := (\bar{T}^s_{\text{concat}})_{s\in S}$ yields a primal feasible solution to the LSP-J whenever this LSP-T is feasible. This primal heuristic can be integrated into Algorithm 1 to obtain an upper bound μ in every node.

3.4. **Phase 1: Collecting Candidate Assignments.** In phase one we collect a set of job to server assignment candidates. The hope is that this set is small enough to solve an LSP-J for each of them by Algorithm 1. This is indeed the case for our benchmark problems. For the candidate selection we use a branch-and-bound algorithm employing similar ideas as the Phase-2 algorithm for the LSP-J: Each node N corresponds to a partial assignment $\bar{J}^s(N)$. By the triangle inequality,

$$\max_{s\in S} \ell(T_{\text{LSP-s}}{}^s(\bar{J}^s(N), d^s)) \tag{21}$$

is a valid lower bound for every extension to a full assignment and, thus, for every extension to a feasible solution. Whenever we reach a leaf we use Remark 2 to improve the current upper bound. Every leaf whose lower bound is below the current best upper bound will be collected as a candidate. The method is given in Algorithm 2.

Algorithm 2 Candidate Selection for the LSP

```
Require: Data of the LSP
Ensure: The set $\Omega$ contains an optimal server assignment $\left(\bar{J}^s\right)_{s \in S}$ of the LSP.
  $\mu \leftarrow \infty$
  $\Omega \leftarrow \emptyset$
  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
      evaluate $T_{\text{LSP-s}}{}^s(\bar{J}^s(N), d^s)$
    end for
    $\lambda(N) \leftarrow \max_{s \in S} \ell\big(T_{\text{LSP-s}}{}^s(\bar{J}^s(N), d^s)\big)$
    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 $\Omega$
        solve LSP-T for the LSP-s tours, adjust $\mu$ if appropriate
      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 $\Omega$
```

Whenever the LSP is feasible, Ω returned by Algorithm 2 contains the server assignment of an optimal solution.

3.5. **Top Level: An Algorithm for the** LSP**.** The top-level method solving the LSP is shown in Algorithm 3: Check all candidates and pick the one with best LSP-J-solution.

4. Computational Results

In this section, we report on comprehensive computational results achieved by our algorithm CBB for the LSP. We mainly compare the quality of dual bounds (rather than running times) to the LP relaxation of an MILP model based on a time-space network (see the appendix for details about the MILP model). Such a model is often used as a basis for devising a column generation algorithm via Dantzig-Wolfe decomposition, a natural competitor of our new method.

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

```
Require: Data of the LSP
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 LSP.
  $\mu \leftarrow \infty$
  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 $(\bar{T}^s_{\text{OPT}}, \bar{t}^s_{\text{OPT}}, \bar{r}^s_{\text{OPT}})_{s\in S}$
```

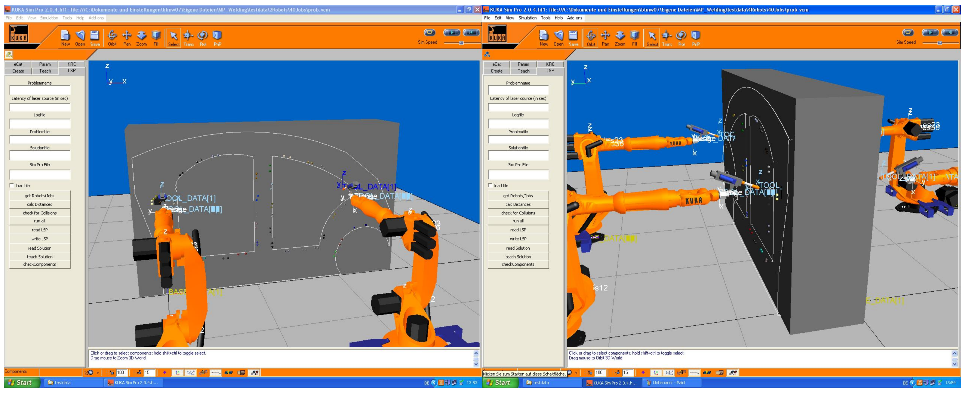

FIGURE 1. Data files with two and four robots

All computations have been done on an Intel Xeon processor with 2.33 Ghz running ubuntu linux 10.04 in 64 bit mode(kernel 2.6.32, gcc version 4.4.3). The system was equipped with 64 gigabyte of memory. The LP relaxation of the time space network model was solved using the primal simplex method of IBM ILOG cplex version 12.2 [7], which was the fastest method for these models. Cplex was also used to solve the LSP-T subproblems in CBB. For the LSP-s subproblems we implemented a branch-and-cut solver in the framework Scip version 2.0 [1].

4.1. **Data Generation.** We created two simulation models. Figure 1 shows the instances for two and four robots with 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; the colors indicate which positions belong to the same jobs.

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 a robot. 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.

4.2. **Performance Metrics.** The yard stick for all test runs was a primal bound generated by our method with a time limit of one day. 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 time space network model run for one day on all instances, but checking just a few instances showed that CBB was the only method to generate primal bounds for every instance in one day per instance. In the test computations we wanted to reveal why CBB performs so well on our test data.

4.3. **Results.** First we prescribe the robot to job assignment of the yard-stick solution resulting in a LSP-J instance. Figures 3 and 5 provide the dual bounds in the root nodes of the respective branch-and-bound trees. The LSP-s relaxation used in CBB provides a stronger bound than the linear programming relaxation of the time-space network model. And it requires less computational efford to evaluate it, as can be seen in Figures 4 and 6. If we do not prescribe a robot to job assignment, the situation looks similar. In Figures 7 and 9 we compare the value of the linear programming relaxation to the value of a routing optimal solution, i.e., the minimal makespan of a solution when resource sharing is ignored. The respective computation times are given in Figures 8 and 10. Note that whenever a value is missing in the chart, this means that CPLEX refused to read the time-space network model because it was too large.

The summarized findings of our extensive computational results on the LSP-J are collected in Figure 11. 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 12 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 introducing collision avoidance does not increase the optimal *makespan* by much. In order to illustrate how different the corresponding optimal *solutions* can be,

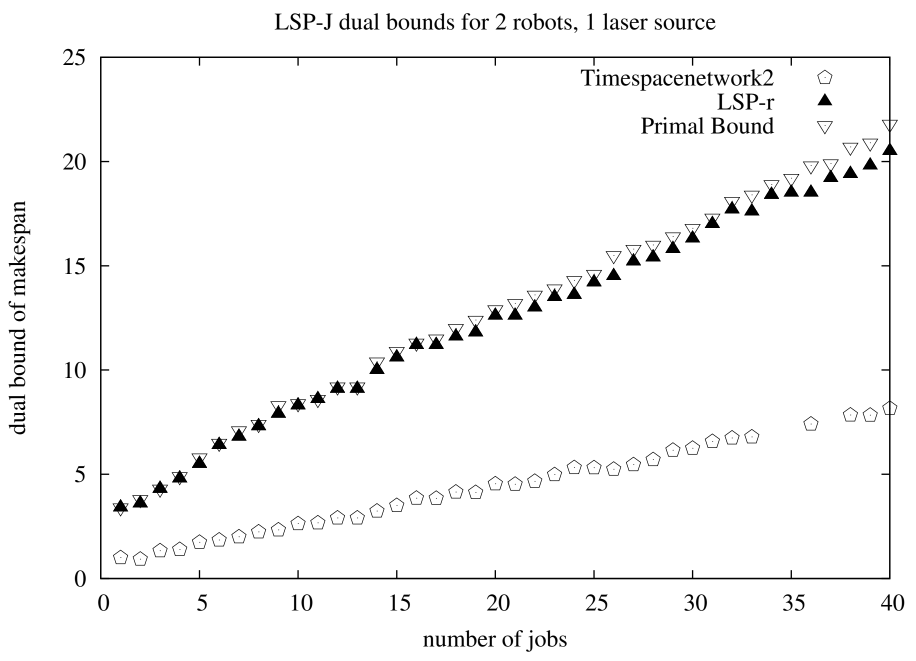

FIGURE 2. Dual bounds for two robots, one laser source (LSP-J)

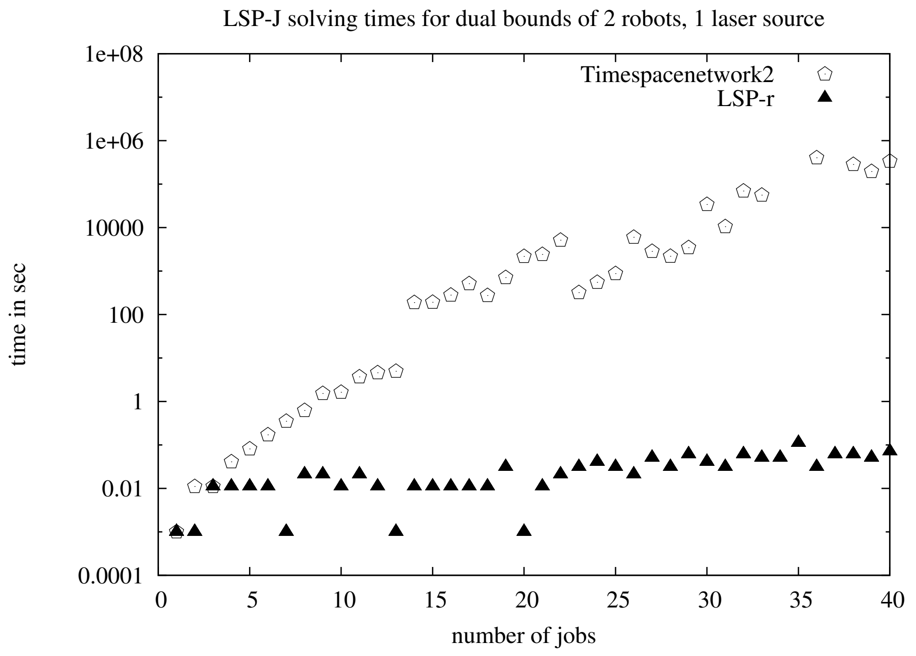

FIGURE 3. CPU times for two robots, one laser source (LSP-J)

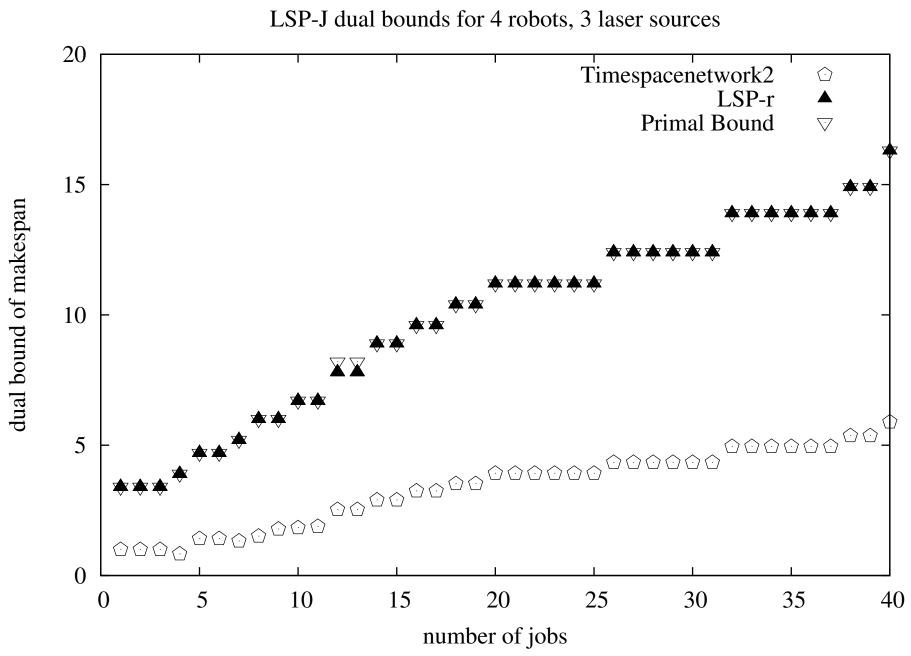

FIGURE 4. Dual bounds for four robots, three laser sources (LSP-J)

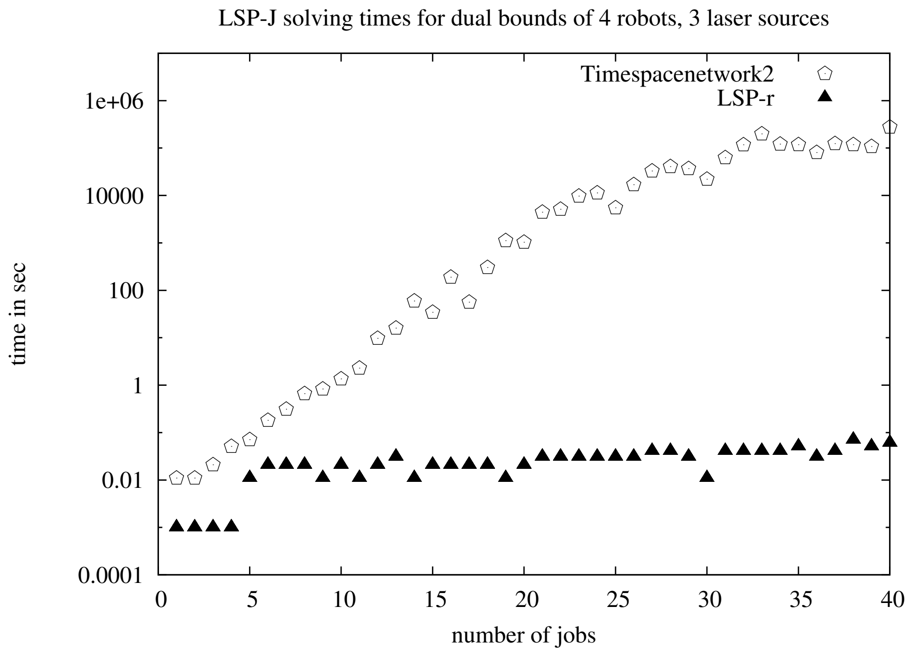

FIGURE 5. CPU times for four robots, three laser sources (LSP-J)

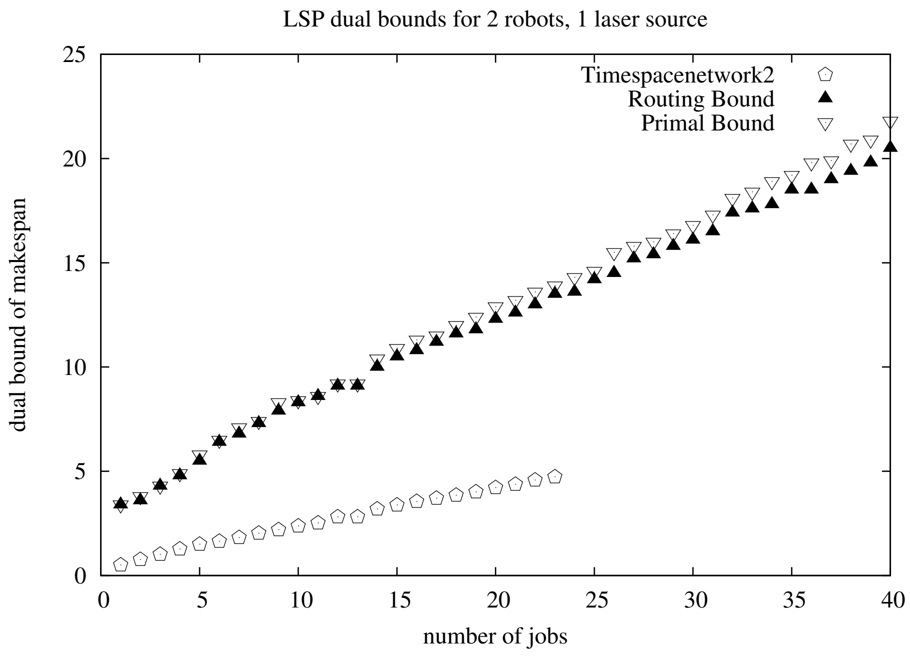

FIGURE 6. Dual bounds for two robots, one laser source (LSP)

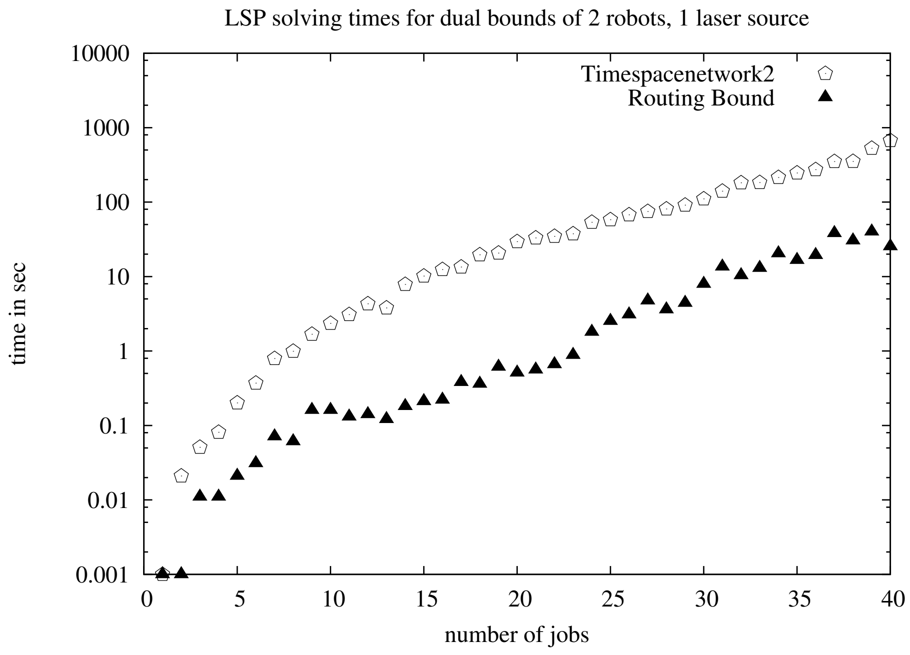

FIGURE 7. CPU times for two robots, one laser source (LSP)

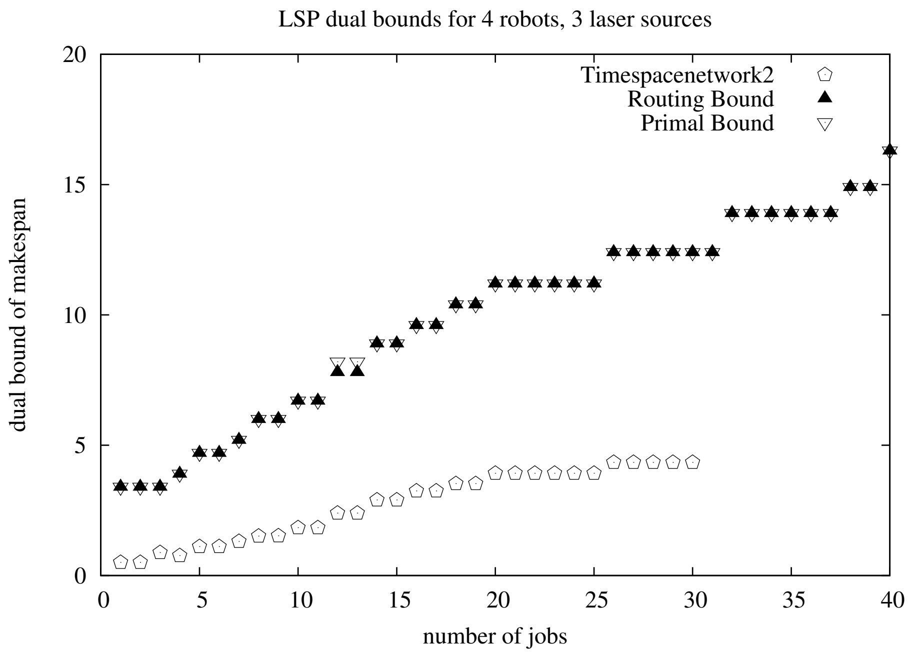

FIGURE 8. Dual bounds for four robots, three laser sources (LSP)

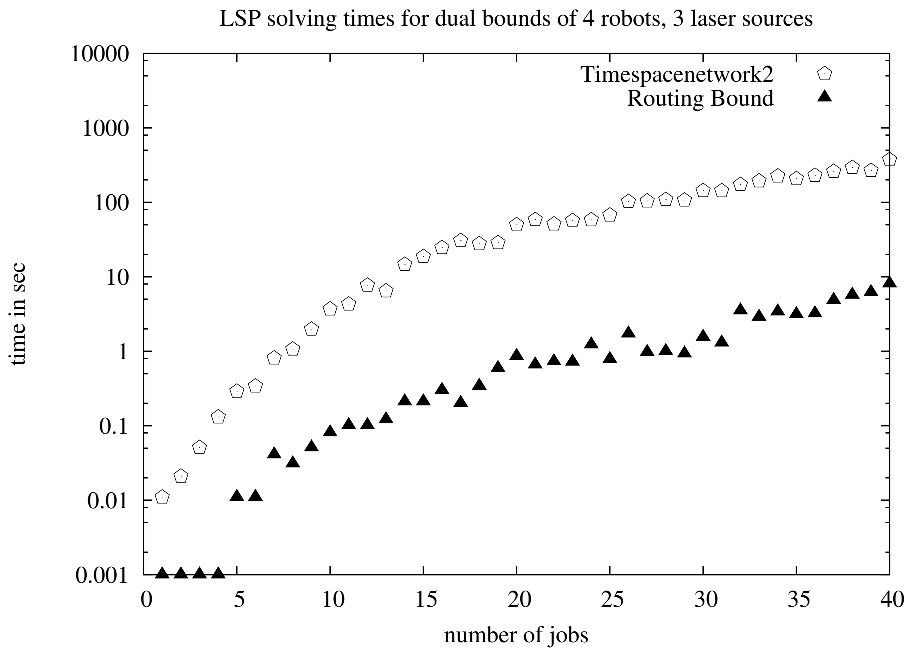

FIGURE 9. CPU times for four robots, three laser sources (LSP)

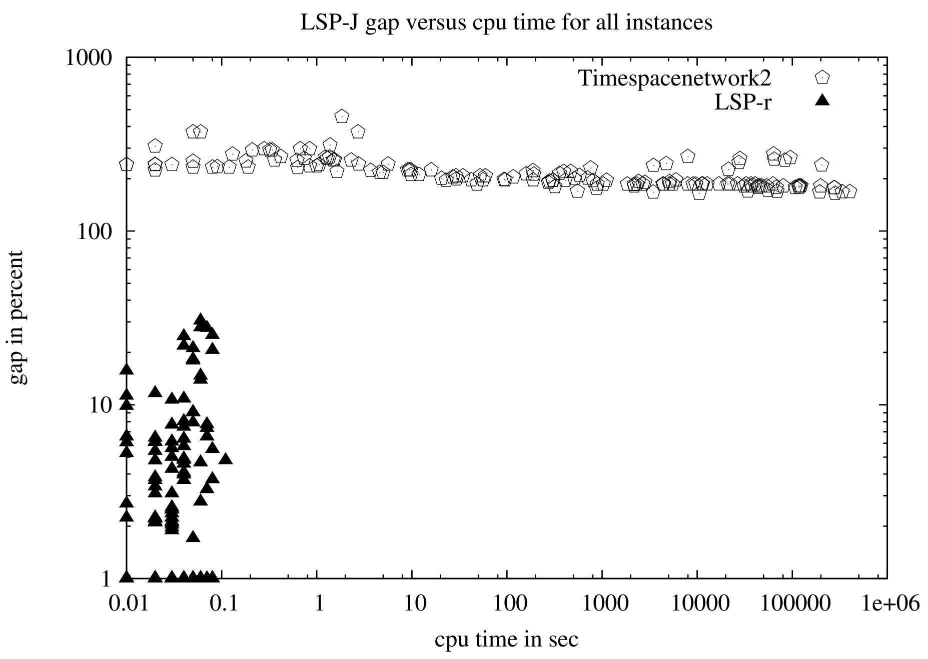

FIGURE 10. Solution times and gaps of all LSP-J instances

we show in Figure 13 a snapshot of an optimal solution without collision avoidance and two snapshots of an optimal solution with collision avoidance. The solution with collision avoidance has a totally different routing. This routing leads to a solution where the robots are never in the central area at the same time, whereas the solution without collision avoidance allows both robots to meet in the center. 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 center. It is very likely that the more conservative solution produced by our concept of collision avoidance would be favored by most engineers.

5. CONCLUSIONS

We introduced the new laser sharing problem LSP arising in car body manufacturing: find a scheduled dispatch with minimal makespan for welding robots that need to share some laser sources and must avoid collisions in the welding cell. This problem combines the challenges "vehicle routing", "scheduling with shared renewable resources", and "makespan minimization". We learned that standard MILP techniques do not work well. However, they work well for the subproblems with only one server or prescribed tours, resp. Our new algorithm CBB is a branch-and-bound algorithm that utilizes optimal solutions to these NP-hard subproblems. State-of-the-art MILP methods can solve the subproblems quite well because industrial-scale instances for the LSP lead to tractable instances for the subproblems. The benefit of solving the diffcult subproblems as opposed to mere LP relaxations is the quality of the resulting dual bounds. We have demonstrated this by solving instances based on data that was generated by robot simulations of an industry-standard tool: Our algorithm CBB is so far the only algorithm that can solve instances to proven

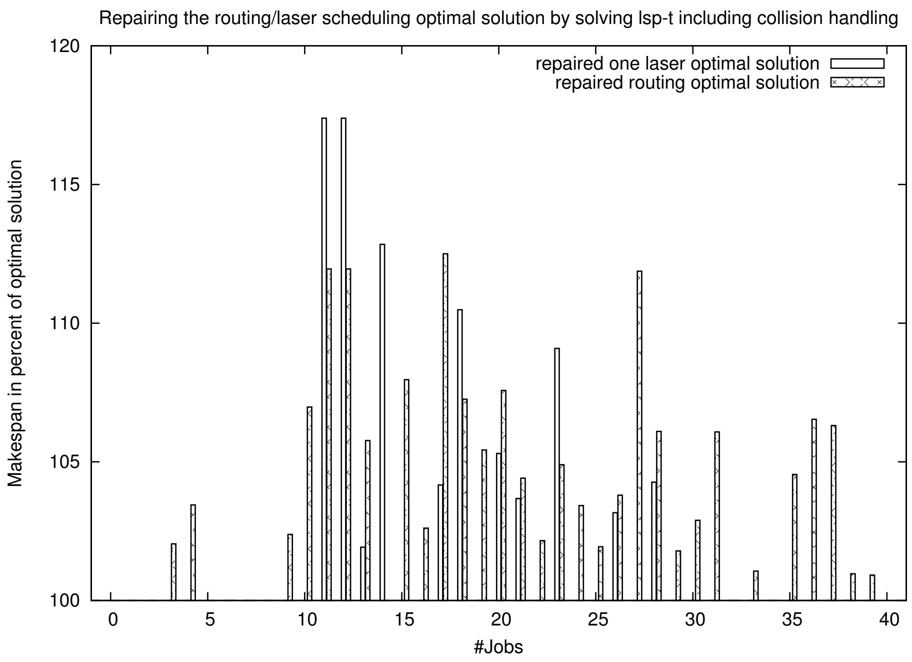

FIGURE 11. Gaps for route-first-schedule-second

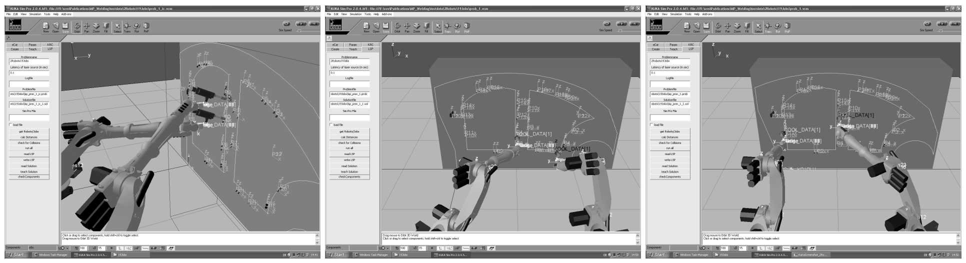

FIGURE 12. Snapshots of an optimal collision-ignoring (left) and an optimal collision-avoiding (middle and right) solution

optimality that consist of up to 40 welding jobs, four robots and three laser sources. We believe that the key ingredients of CBB may also help to design successful algorithms for other combined routing and scheduling problems. With our tool, the original application problem can now be answered: the question how many laser sources are really needed in a welding cell. Since the laser sources are much more expensive than the welding robots, the correct answer to this can be worth big money in planning new production lines for car body shops.

REFERENCES

[1] T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.
[2] P. Brucker and S. Knust. *Complex Scheduling*. Springer-Verlag, Berlin, Heidelberg, New York, 2006.

[3] J. Desroisers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constraint routing and scheduling. In M. Ball, T.L. Magnanti, C.L. Monma, and G. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35–140. Elsevier, Amsterdam, 1995.
[4] 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.
[5] 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.
[6] Christoph Hempsch and Stefan Irnich. Vehicle routing problems with inter-tour resource constraints. Golden, Bruce (ed.) et al., The vehicle routing problem. Latest advances and new challenges. New York, NY: Springer. Operations Research/Computer Science Interfaces Series 43, 421-444 (2008)., 2008.
[7] IBM. *CPLEX Mathematical Programming Software*. information online available[1].
[8] Stefan Irnich. Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008.
[9] Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation*, pages 33–65. Springer US, 2005.
[10] Sigrid Knust. *Shop-Scheduling Problems with Transportation*. PhD thesis, Universiät Osnabrück, 2000.
[11] 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.
[12] J. Rambau and C. Schwarz. How to avoid collisions in scheduling industrial robots? Preprint, Universität Bayreuth, 2010.
[13] C. Schwarz and J. Rambau. Optimal dispatching of welding robots. Slides for the 13th Combinatorial Optimization Workshop, January 2009.
[14] Paolo (ed.) Toth and Daniele (ed.) Vigo. *The vehicle routing problem.* SIAM Monographs on Discrete Mathematics and Applications. 9. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics. xviii, 367 p. , 2002.
[15] W. Welz. Route planning for robot systems. Diplomarbeit, Technische Universität Berlin, 2010.
[16] W. Welz and M. Skutella. Route planning for robot systems. Talk WD-07.1 of the International Conference Operations Research, 2010.

Appendix A. A Mixed Integer Linear Model

Our comparision 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 makespan objective alone causes large ingegrality gaps in the master problem that can not be reduced by integral solutions to the pricing problem alone. Moreover, all constraints like resource sharing involving more than one server must enter the master problem as well, leading to even larger integrality gaps. Moreover, the time-space network model time requires a discretizion of time, and the corresponding discretization error is not neglectable: Rounding up all driving, waiting, and switching times in a dispatch may add up to a substantial over-estimation of its length, mainly because all tours need synchronization. In our tests, rounding errors w.r.t. a step-size of $0.1\,\mathrm{s}$ sometimes add up to $10\,\%$ of the optimal makespan.

Nevertheless, we provide the model that we used. 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 $G^{s,l} = (V^{s,l}, A^{s,l})$ be the network for Robot s and Laser Source l with node set

$$V^{s,l} := \{(t,q) \mid 0 \leq t \leq \mu, j \in J^r, q \in \{j_a, j_b\} \text{ or } q = d^r\}$$

At node (t_1,q_1) Robot s has two possibilities: It can wait one time unit, represented as a move to node (t_1+1,q_1) in the time-space network, or it can move from Position q_1 to

[1] http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/

Position q_2, represented by a move to node $(t_1+\delta^s(q_1,q_2),q_2)$ in the time-space network. When both positions belong to the same job then this move is done in welding mode otherwise in driving mode. Thus the arc set is

$$\begin{aligned}A^{s,l} := \{(t_1,q_1,t_2,q_2) \in V^{s,l} \times V^{s,l} \mid\ & q_1 = q_2 \wedge t_2 = t_1+1\\ & \vee q_1 \neq q_2 \wedge t_2 = \delta^r(q_1,q_2)\}\\ & \cup \{(\mu,d^s,0,d^s)\}\end{aligned}$$

The tour $\bar{T}^s$ of Server s is modeled as a binary flow in $A^{s,l}$. For this, we use variables $y_a^{s,l} \in \{0,1\}$ where l is the connected laser source. The first arc in the tour selects the network and, thus, the laser source. Resource conflicts are prevented by a set-packing constraint over all conflicting arcs for each time position $t = 0,\dots,\mu$ and each resource. More specifically: Define by $C^{l,t}$ be the set of all welding arcs which cover time position t, i.e., $(s,t_1,q_1,t_2,q_2) \in C^{l,t}$ if $(q_1,q_2) \in \{(j_a,j_b),(j_b,j_a)\} =: P^j$ and $t_1 \le t \le t_1+\delta^s(q_1,q_2)+\tau^l-1$.

The resulting model is:

Problem 3 (Timespacenetwork).

$$\min z$$

subject to

$$\sum_{l\in L}\sum_{a\in A} t_2 y_a^{s,l} \le z \qquad \forall\, s\in S, \tag{22}$$

$$A := \{(t_1,q_1,t_2,d^r) \in A^{s,l} \mid q_1 \neq d^s\}$$

$$\sum_{(v,w)\in A^{s,l}} y^{s,l}_{(v,w)} - \sum_{(w,v)\in A^{s,l}} y^{s,l}_{(w,v)} = 0 \qquad \forall\, s\in S, l\in L, v\in V^{s,l} \tag{23}$$

$$\sum_{l\in L}\sum_{a\in A} y_a^{s,l} \le 1 \qquad s\in S, A := \{(0,d^s,t_2,q)\in A^{s,l}\} \tag{24}$$

$$\sum_{(s,a)\in C^{l,t}} y_a^{s,l} \le 1 \qquad \forall\, l\in L, t=0,\dots,\mu \tag{25}$$

$$\sum_{l\in L}\sum_{a\in A_s} y_a^{s,l} + \sum_{l\in L}\sum_{a\in A_s} y_a^{s,l} \le 1 \qquad \forall\,(s,q_1^s,q_2^s,s',q_1^{s'},q_2^{s'}) \in C_{\mathrm{ll}} \tag{26}$$

$$A_s := \{(t_1,q_1^s,t_2,q_2^s) \in A^{s,l} : t_1 \le t \le t_2\}$$
$$A_{s'} := \{(t_1,q_1^{s'},t_2,q_2^{s'}) \in A^{s',l} : t_1 \le t \le t_2\}$$
$$t = 0,\dots,\mu$$

$$\sum_{l\in L}\sum_{a\in A} y_a^{s,l} + \sum_{l\in L} y^{s',l}_{(t,q^{s'},t+1,q^{s'})} \le 1 \qquad \forall\,(s,q_1^s,q_2^s,s',q^{s'}) \in C_{\mathrm{lp}}, \tag{27}$$

$$t = 0,\dots,\mu$$
$$A := \{(t_1,q_1^s,t_2,q_2^s) \in A^{s,l} : t_1 \le t \le t_2\}$$

$$\sum_{l\in L}\sum_{s\in S: j\in J^s}\sum_{a\in A} y_a^{s,l} = 1 \qquad \forall\, j\in J \tag{28}$$

$$A := \{(t_1,q_1,t_2,q_2) \in A^{s,l} : (q_1,q_2) \in P^j\}$$

$$\sum_{l\in L}\sum_{r\in R}\sum_{a\in A} y_a^{s,l} \le 1 \qquad \forall\, q_1 \in \bigcup_{s\in S}\{d^s\} \cup \bigcup_{j\in J}\{j_a,j_b\} \tag{29}$$

$$A := \{(t_1,q_1,t_2,q_2) \in A^{s,l} : q_1 \neq q_2\}$$

$$y_a^{s,l} \in \{0,1\} \quad \forall\, s\in S, l\in L, a\in A^{s,l} \tag{30}$$

$$z \in \mathbb{Z}_{\geq 0} \tag{31}$$

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