How to avoid collisions in scheduling industrial robots?

Jörg Rambau and Cornelius Schwarz

1 Introduction

If industrial robots are working in the same area, collisions must be avoided. Usually this is done by hand: distribute the jobs among the robots in such a way that no collision is possible. To the best of our knowledge, no automatic procedure is known that automatically assigns jobs to robots and routes robots collision-free in one step.

Our new approach integrates collision avoidance into the optimization process. To this end, we model a possible collision as a shared resource among the robots. When Robot r moves into a critical area it tries to acquire all needed collision resources. As long as there are other robots crossing the paths of r, at least one of these resources is in use. This causes r to wait until the critical area is free.

We present our collision concept for the *Laser Source Sharing Problem (LSP)* [Grötschel et al (2006), Rambau and Schwarz (2009)]: dispatch arc welding robots (laser welding technology) with as few laser sources as possible. Here another resource plays an important role: Every robot needs to be connected to a laser source supplying it with the necessary energy. While a laser source can switch between robots, it can supply only one robot at a time. Because the main restriction in production scheduling is the cycle time, our objective is to minimize the makespan.

The LSP without collision avoidance was studied in [Grötschel et al (2006)] for fixed tours. In [Schneider (2006)], routing was integrated. Practical problem sizes could be handled exactly for the first time in [Rambau and Schwarz (2009)], based on bounds from combinatorial ATSP-relaxations.

Here, first, we integrate the handling of collisions directly in the optimization algorithm and, second, we present a benchmark suite of realistic test data generated with the industrial robot simulation software KuKa SimPro [KukaSimPro (2010)]. The new collision-aware algorithm outperforms other approaches on this benchmark suite (including [Rambau and Schwarz (2009)] if there are at most three robots).

Jörg Rambau
University of Bayreuth, Bayreuth, Germany, e-mail: joerg.rambau@uni-bayreuth.de

Cornelius Schwarz
University of Bayreuth, Bayreuth, Germany e-mail: cornelius.schwarz@uni-bayreuth.de

2 Problem definition – a new model for collisions

Let R be a set of robots, J a set of jobs and L a set of laser sources. Each robot $r \in R$ has a nullposition o_r, where the tour has to start and to end. Each job $j \in J$ has two end positions j_a, j_b. If the service of a job starts at j_a it has to finish at j_b, and vice versa. With $Q := \{o_r \mid r \in R\} \cup \{j_a, j_b \mid j \in J\}$ we refer to the set of all positions. The driving time of Robot r from Positions q_i to q_j is denoted by $\delta_r(q_i, q_j)$. Job processing times are special driving times $\delta_r(j_a, j_b)$, $\delta_r(j_b, j_a)$ depending on the actual direction. Switching Laser Source l incurs a delay of δ_l.

We distinguish between two types of collisions: *line-line* collisions $C_{\mathrm{ll}} \subseteq R \times Q \times Q \times R \times Q \times Q$ and *line-point* collisions $C_{\mathrm{lp}} \subseteq R \times Q \times Q \times R \times Q$. An entry $(r_1, p_1, q_1, r_2, p_2, q_2) \in C_{\mathrm{ll}}$ is interpreted as follows: There exists two start times t_1, t_2 with the following property: when r_i starts at time t_i to move from p_i to q_i, there will be a collision between r_1 and r_2. Our (conservative) rule is: we do not allow these two robot moves to overlap in time. A line-point collision $(r_1, p_1, q_1, r_2, p_2) \in C_{\mathrm{lp}}$ means that Robot r_1 moving from p_1 to q_2 will hit Robot r_2 waiting at p_2. Therefore, Robot r_1 is not allowed to perform this move until Robot r_2 has left Position p_2.

The task is to assign each $j \in J$ to a robot $r \in R$, each robot $r \in R$ to a laser source $l \in L$, and to create a scheduled tour for every robot through all assigned jobs so that

- each job is assigned to exactly one robot,
- each robot is assigned to exactly one laser source,
- jobs assigned to robots sharing a laser source do not overlap in time,
- all robot moves are collision free.

The cost of a scheduled tour is the time length, i.e., the time when the robot finishes its tour at o_r. The goal is to minimize the makespan, which is the maximum over the tour costs. If we restrict to one robot (the *1-server problem*) and set for all jobs $j_a = j_b, \delta_r(j_a, j_b) = \delta_r(j_b, j_a) = 0$ we get an ATSP, which is NP-hard. Thus, the laser sharing problem is also NP-hard.

3 The algorithm – how to avoid collisions

Assume that an assignment $\mathfrak{J} : R \to 2^J$ of robots to jobs is fixed. The resulting problem is called *LSP-J*. If resources are neglected, then we can solve all 1-server problems separately: Let $G = (V, A)$ be a complete digraph with node set $V := \{o_r\} \cup \{j_a, j_b \mid j \in \mathfrak{J}(r)\}$. To force the processing of Job j, we introduce a new vertex j_c between j_a and j_b. This way we end up with an ATSP which can be solved by any exact solver, for instance [Fischetti et al (2003)].

We will denote by $t_r^{\text{LSP-r}}(K, q) := \text{LSP-r}_r(K, q)$ a call to an exact algorithm solving the one server problem for Robot r starting at position q, processing all jobs in set K and ending at o_r. The resulting tour is saved in $t_r^{\text{LSP-r}}(K, q)$.

Tuchscherer et al. [Grötschel et al (2006)] proposed a mixed integer model for the case where in advance the tours t_r are known. We extend this model to handle collisions as well: For each position q of t_r, let the variable $x_q \geq 0$ denote the starting time of the process at q. Depending on q this is either a welding or a driving task. For each two positions q, p of *different* robots, define a variable $y_{p,q} \in \{0,1\}$ with the meaning: p is started before q if and only if $y_{p,q} = 0$. The variables $u_{r,l} \in \{0,1\}$ describe the assignments of robots to laser sources. This leads to the following mixed integer model:

Problem 1 (LSP-T).

$$\min z \tag{1}$$

subject to

$$\begin{aligned}
x_{q_{|t_r|-1}} + \delta_r(q_{|t_r|-1}, o_r) &\leq z && \forall\, r \in R && (2)\\
x_{q_i} + \delta_r(q_i, q_{i+1}) &\leq x_{q_{i+1}} && \forall\, r \in R, i = 1,\dots,|t_r|-2 && (3)\\
x_{p_i} + \delta_r(p_i, p_{i+1}) - x_{q_j} &\leq M(1 - y_{p_i,q_j}) && \forall\, r,s \in R, r \neq s, &&\\
&&& i = 1,\dots,|t_r|-1, j = 1,\dots,|t_s|-1 &&\\
&&& (r, p_i, p_{i+1}, s, q_j, q_{j+1}) \in C_{\text{ll}} && (4)\\
y_{p_i,q_{j-1}} - y_{p_i,q_j} &= 0 && \forall\, r,s \in R, r \neq s &&\\
&&& i = 1,\dots,|t_r|-1, j = 2,\dots,|t_s|-1 &&\\
&&& (r, p_i, p_{i+1}, s, q_j) \in C_{\text{lp}} && (5)\\
y_{p,q} + y_{q,p} &= 1 && \forall\, r,s \in R, s \neq r, p \in t_r, q \in t_s && (6)\\
x_{p_i} + \delta_r(p_i, p_{i+1}) + \delta - x_{q_j} &\leq M(3 - u_{l,r} && \forall\, r,s \in R, r \neq s, &&\\
&\qquad - u_{l,s} - y_{p_i,q_j}) && i = 1,\dots,|t_r|-1, j = 1,\dots,|t_s|-1, &&\\
&&& j, i \mod 2 = 0,\ \forall\, l \in L && (7)\\
\sum_{l \in L} u_{l,r} &= 1 && \forall\, l \in L && (8)\\
x &\geq 0 &&&& (9)\\
z &\geq 0 &&&& (10)\\
y_{p,q} &\in 0,1 && \forall\, r,s \in R, r \neq s, p \in t_r, q \in t_s && (11)
\end{aligned}$$

Line-line collisions are handled with the big-M constraint (4). Whenever the task starting at p and the one starting at q are in conflict then the first one has to finish before the other one can start. Due to (6) one task is always marked as the first one.

If $(r, p_i, p_{i+1}, s, q_j)$ is a line-point collision then s has to leave q_j before r starts at p_i or to enter q_j after r reached p_{i+1}. This is handled in (5).

Let t_r be a tour for Robot r, $J(t_r)$ the jobs processed in t_r, and $\ell(t_r)$ the time length then for every given tour set $T := (t_r)_{r \in R}$ we can solve LSP-T with a mixed integer solver and obtain an optimal collision free solution. Indeed, this is the part where collisions are handled within the following branch-and-bound algorithm: Let t_r be a partial tour, i. e., $J(t_r) \subsetneq \mathcal{J}^{-1}(r)$, then an optimal collision free scheduling of T yields a lower bound for every extension of T to a feasible solution of LSP-J. Given an estimate τ_r how long Robot r needs at least for processing all remaining jobs and returning home when starting at the end position q_r of t_r we can strengthen this bound by temporally replacing $\delta_r(q_r, o_r) := \tau_r$. With $T^{\text{LSP-T}}(T, \tau) := \text{LSP-T}(T, \tau)$

we will denote a call to an exact solver storing the optimal schedule of LSP-T in $T^{\text{LSP-T}}(T,\tau)$.

We use $\tau_r := \ell\big(t_r^{\text{LSP-r}}(K_r,q_r)\big)$ with $K_r := \mathfrak{J}^{-1}(r) \setminus J(t_r)$. This can be interpreted as completing the tour in an optimal manner relaxing the resource constraints. Now we can solve LSP-J using B&B with a node for each $(t_r)_{r\in R}$ and child nodes corresponding to all single-job extensions of a single t_r.

We summarize the algorithm for LSP-J: For $r \in R$, $T := (t_r)_{r\in R}$ keeps a partial tour t_r for every robot. We write μ for the best upper bound and λ for the lower bound of the current node:

Algorithm 1 (Combinatorial Branch-and-Bound for LSP-J)

```
   INPUT: Data of LSP-J
  OUTPUT: A set $T_{\text{OPT}}$ with optimal scheduled tours $(t_r)_{r\in R}$.

1. initialize: $t_r := ()$ for all $r \in R$
2. set $\mu := \infty$, $T_{\text{OPT}} := T := \big(()\big)_{r\in R}$ // empty tours
3. CBB($T$)
4. return $T_{\text{OPT}}$

Procedure CBB($T$) for $T = (t_r)_{r\in R}$:

1. for all $r \in R$,
   set $q_r$ to the last position of $t_r$ or $o_r$ if $t_r = ()$
   set $K_r := J(r) \setminus J(t_r)$
   calculate $t_r^{\text{LSP-r}}(K_r,q_r) := \text{LSP-r}_r(K_r,q_r), \tau_r := \ell(t_r^{\text{LSP-r}}(K_r,q_r))$
2. set $\tau := (\tau_r)_{r\in R}$ and calculate
   $T^{\text{LSP-T}}(T,\tau) := \text{LSP-T}(T,\tau)$,
   $\lambda := \ell(T^{\text{LSP-T}}(T,\tau))$ // lower bound
3. if $\lambda > \mu$ return // pruning
4. if $J(t_r) = J(r)$ for all $r \in R$ // all jobs are ordered

   a. complete the tours, i.e., append $o_r$ to $t_r$ $\forall$ $r \in R$
   b. evaluate the current solution:
      set $\tau_r := \delta_r(q_r,o_r)$, $r \in R$, $\tau := (\tau_r)_{r\in R}$
      calculate $T_{\text{new}} := \text{LSP-T}(T,\tau)$
   c. if $\ell(T_{\text{new}}) < \mu$, set $T_{\text{OPT}} := T_{\text{new}}$, $\mu := \ell(T_{\text{new}})$ // new best solution
   d. return

5. choose $r \in R$ with $K_r \neq \emptyset$
6. for all $j \in K_r$, $(q_{\text{start}},q_{\text{end}}) \in \{(j_a,j_b),(j_b,j_a)\}$ // iterate over unscheduled jobs

   a. append $(q_{\text{start}},q_{\text{end}})$ to $t_r$ // Job $j$ is welded next from $q_{\text{start}}$ to $q_{\text{end}}$
   b. CBB($T$) // recurse
   c. remove $(q_{\text{start}},q_{\text{end}})$ from $t_r$ // backtrack
```

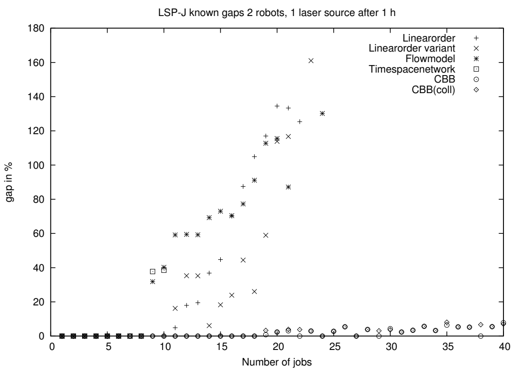
Remark 1 *In [Rambau and Schwarz (2009)] we proposed a similar algorithm where collision avoidance had not been taken into account. In the branching step a job to be processed next by the laser source was chosen. Our new algorithm only selects the next job on some robot. So we reduced the search tree at the costs of solving an additional MIP at each iteration. This reduced running times by up to* 90%.

It remains to find the assignment $\mathfrak{J}$ of an optimal solution. We reduce the number of all possible assignments to a usual small set of promising ones that must contain an optimal one and enumerate. See [Rambau and Schwarz (2009)] for details.

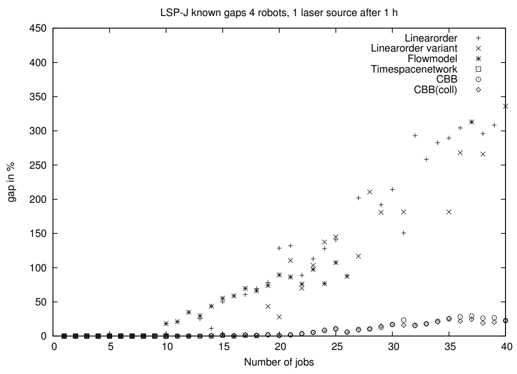
4 Computational Results on Realistic Data

In a first step we compare our algorithm to four mixed integer models of LSP: The linear ordering based model of Schneider [Schneider (2006)], a variation of it with fewer big-M constraints, a network model with nodes (r, j, d, k, l) with the meaning: Job j is processed by Robot r in Direction d as the k-th job of Laser Source l, and a time-space network model, as often used for column generation methods.

We generated our testdata using the robot simulation software KuKa Sim Pro 2.0 [KukaSimPro (2010)]. In order to compare with the time space network model, all distances have been rounded with precision 10^{-1}. None of the MIP models can take care of collision avoidance. For a fair comparison, we ran two version of our branch-and-bound algorithm: with collisions ignored (CBB) and with collision avoidance CBB(coll). Figure 1 shows the known gaps of 40 instances with 2 and 4 robots. All



(a) Two robots



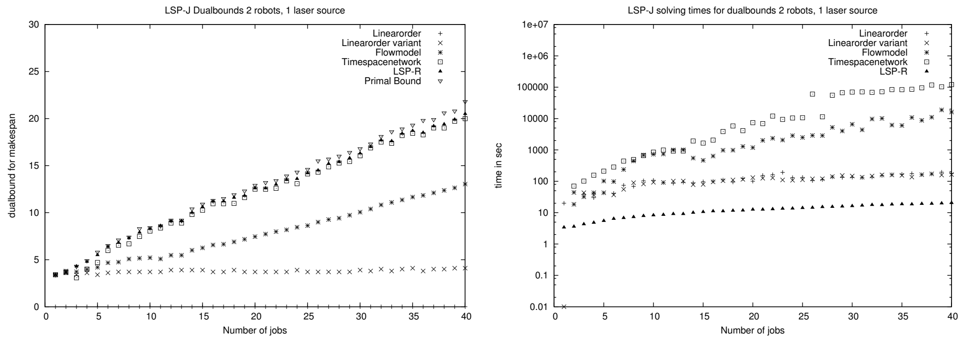
(b) Four robots

Fig. 1: Gaps achieved by the algorithms for LSP-J after 1h

calculations were performed on an Intel Xeon E 5410 cpu, 2.33Ghz, 64 GB RAM, ubuntu linux 8.04, and cplex 12.1 (default options + barrier for root LPs).

In the 2-robot case all mixed integer models could be solved for up to 8 jobs to optimality. For more robots the gap increases until cplex was not able to find any primal solution. This is the case at about 23 jobs. With 4 robots the models seems to perform better but take into account that 40 jobs assigned to 4 robots means every robot has only to do about 10 jobs. The combinatorial branch-and-bound method was able to solve up to 20 jobs to optimality in 1 h and obtained small gaps for all bigger instances. This remains true also with collision handling.

One reason for the good performance of our code is the dualbound. Figure 2 compares our dualbound obtained by solving an LSP-r for every robot with the root LP relaxation of the mixed integer models. Solving LSP-r provides the strongest dual bounds in the shortest computation times.



(a) Dualbounds (b) Computation times

Fig. 2: Dualbounds of the algorithms in root node (two robots, one laser source)

5 Conclusions

We have shown that the successful combinatorial branch and bound technique in [Rambau and Schwarz (2009)] using lower-bounds by solving NP-hard but small subproblems remains successful when collisions are to be avoided. We verified this on a realistic benchmark suite created with industry standard simulation software. Whereas it is quite intricate to even incorporate both collision avoidance and laser source sharing into the competing models based on LP-relaxations for makespan minimization, our branch-and-bound algorithm based on combinatorial relaxations is able to solve the full problem for industrial scales in reasonable time.

References

[Fischetti et al (2003)] Fischetti M, Lodi A, Toth P, (2003) Solving real-world ATSP instances by branch-and-cut. Jünger, Michael (ed.) et al., Combinatorial optimization - Eureka, you shrink. Papers dedicated to Jack Edmonds. 5th international workshop, Aussois, France, March 5-9, 2001. Revised papers. Berlin: Springer. Lect. Notes Comput. Sci. 2570, 64-77 (2003).

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

[KukaSimPro (2010)] KukaSimPro, (2010) KuKa SimPro 2.0. KuKa, information available [1]

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

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

[1] http://www.kuka-robotics.com/en/products/software/kuka_sim