A generalized job-shop problem with more than one resource demand per task

Joachim Schauer* Cornelius Schwarz**

Abstract

We study a generalized job-shop problem called the *Laser Sharing Problem with fixed tours (LSP-T)* where the tasks may need more than one resource simultaneously. This fact will be used to model possible collisions between industrial robots. For three robots we will show that the special case where only one resource is used by more than one robot is already $\mathcal{NP}$-hard. This also implies that one machine scheduling with chained min delay precedence constraints is $\mathcal{NP}$-hard for at least three chains. On the positive side, we present a polynomial algorithm for the two robot case and a pseudo-polynomial algorithm together with an FPTAS for an arbitrary but constant number of robots. This gives a sharp boundary of the complexity status for a constant number of robots.

Keywords: job-shop, FPTAS, complexity, transversal graph

1 Introduction

Over the years shop scheduling problems have achieved much attention in the scheduling literature. In a *Shop Scheduling Problem* one is given *machines* $M_1, \ldots, M_m$ and *jobs* $J_1, \ldots, J_k$. Each job J_i consists of n_i *tasks* $O_{i,j}$. Every task $O_{i,j}$ has to be performed $p_{i,j}$ time units on a dedicated machine $\mu_{i,j}$. In addition in *Job Shop Scheduling* the order of the tasks of every job is prescribed, i. e. we have precedence constraints of the form $O_{i,j} \to O_{i,j+1}$ (cf. e.g. Pinedo [9]). We will focus on the makespan objective $C_{\max}$ which corresponds to the completion time of the last job.

The real world application behind this article is the *Laser Sharing Problem with fixed Tours (LSP-T)* which arises in car body shops: Let $R = \{R_1, \ldots, R_k\}$ be a set of arc welding *robots*. Every robot R_i has to perform n_i *welding tasks* $W_{i,j}$. To supply them with the necessary energy, every robot has to be connected to a *laser source* $l \in L$. Each laser source can only serve one robot

**University of Bayreuth, Chair of Business Mathematics, D-95440 Bayreuth, Germany, cornelius.schwarz@uni-bayreuth.de

*University of Graz, Department of Statistics and Operations Research, Universitaetsstr. 15, A-8010 Graz, Austria, joachim.schauer@uni-graz.at

at a time, i. e. welding tasks of robots assigned to identical laser sources cannot be processed in parallel. After finishing the current welding task the robot has to perform a transversal move, called *driving task*, $D_{i,j}$ to reach the start position of its next welding task. Thus the tasks of robot R_i are of the form $(O_{i,1}, \ldots, O_{i,2n_i+1}) = (D_{i,1}, W_{i,1}, D_{i,2}, W_{i,2}, \ldots, D_{i,n_i-1}, W_{i,n_i}, D_{i,n_i+1})$.

When the assignment of laser sources to robots is given, the problem described so far can be viewed as job-shop problem with a very special structure: Every robot, i. e. job in job-shop, has a machine solely used by itself to perform the driving tasks and we have one machine for every laser source for the welding tasks. For $|L| = 1$ this is equivalent to $(1 \mid \text{chains}(l_{i,j}) \mid C_{\max})$. Wikum et al. [16] showed that this problem is strongly $\mathcal{NP}$-hard even if the number of tasks per chain is bounded. Note that this result extends to *LSP-T* where the number of robots is part of the input. However, when all processing times as well as all time lags are identical, $(1 \mid \text{chains}(l), p_j = p \mid C_{\max})$, this becomes polynomial time solveable (cf. Munier and Sourd [8]). In his PhD thesis Wikum [15] also proposed a pseudo-polynomial algorithm for the case where only two chains are present.

In the *LSP-T* two additional parts generalize the job-shop structure. First, whenever a laser source l performs two tasks of two different robots consecutively a latency time of τ^l must pass in between. Second, robots working in the same area of the car body might collide - which has to be avoided. To incorporate the latter generalization we use the following model introduced by Rambau and Schwarz [12]: Let q_i respective $q_{i'}$ be two positions such that robot R_i at q_i will collide with robot $R_{i'}$ at $q_{i'}$. If R_i bypasses q_i processing task $O_{i,j}$ and $R_{i'}$ bypasses $q_{i'}$ processing $O_{i',j'}$ then the pair $(O_{i,j}, O_{i',j'})$ is called a *line-line collision*.

Obviously considering only moving robots is not enough to guarantee the absence of collisions. If robot R_i processing task $O_{i,j}$ collides with robot $R_{i'}$ residing at the end position of task $O_{i',j'}$ then this is called a *line-point collision*. We will consider the end and start positions of tasks as part of it, therefore a line-point collisions also implies two line-line collisions. In the above case these are $(O_{i,j}, O_{i',j'})$ and $(O_{i,j}, O_{i',j'+1})$.

Every job-shop instance I can be written as a *LSP-T* instance J in the following way: Every job corresponds to a robot and gets assigned a laser source on its own. Every task of I is mapped to a welding task of J with driving tasks of zero length between. For every pair of task processed on the same machine in I a line-line collision is introduced in J. These collisions can be modeled by a *conflict graph* $G = (V, E)$ with vertex set $V := \{O_{i,j}, i = 1, \ldots, k, j = 1, \ldots, n_i\}$. An edge in this graph between a pair of tasks indicates that they cannot be processed in parallel. By this representation job-shop instances are characterized by conflict graphs G consisting of disjoint cliques, one for each machine. Note however that for general *LSP-T* instances the corresponding conflict graph structure might be more complicated.

1.1 Main Contributions and Related Work

We will fully settle the complexity status of *LSP-T* with a constant number of robots. We will show that the problem is already $\mathcal{NP}$-hard for three robots and one laser source without any collisions and latency. On the positive side we will provide a polynomial time algorithm for two robots based on the geometric approach to job-shop due to Brucker [3]. This provides a first sharp boundary. Moreover for a constant number of robots we will give a pseudo-polynomial time algorithm by using a transversal graph approach introduced to scheduling theory by Middendorf and Timkovsky [7]. This algorithm will be used for deriving an FPTAS by applying a standard rounding argument. Taking this together with the strong $\mathcal{NP}$-hardness result of Wikum et al. [16] from an approximation point of view the derived FPTAS is the best result we can hope for.

The laser sharing problem without collision avoidance has been considered by Grötschel et al. [5] for the first time by providing a mixed integer formulation. Schneider [13] extended this formulation to the general Laser Sharing Problem (LSP). Here in addition to *LSP-T* the welding tasks have to be assigned to robots and for every robot we need to find a sequence through all assigned tasks, i. e. a tour. A branch-and-bound algorithm for the *LSP* can be found in Rambau and Schwarz [10]. Collision handling was added in Rambau and Schwarz [12]. An extensive version including a pseudo-polynomial time algorithm for *LSP-T* with a fixed number of collisions can be found in Rambau and Schwarz [11].

Section 2 introduces the notation of *LSP-T*. In Section 3 we will describe the polynomial algorithm for the two robot case. Section 4 covers the proof of the $\mathcal{NP}$-hardness for three robots, no collision and no latency. The pseudo-polynomial algorithm and the fully polynomial time approximation scheme (FPTAS) is the subject of Section 5.

2 Problem Specification

An instance of *LSP-T* consists of k robots $R_1, \ldots, R_k$, h laser sources $L_1, \ldots, L_h$, n_i welding tasks $W_{i,j}$ $(i = 1, \ldots, k,\ j = 1, \ldots, n_i)$, $n_i + 1$ driving tasks $D_{i,j}$ $(i = 1, \ldots, k, j = 1, \ldots, n_i + 1)$ and sets C_{ll} of line-line collisions, C_{lp} of line-point collisions.

Each robot i has a depot d_i accossiated. Each welding- as well as driving task has assigned a start position and an end position. The end position of task $O_{i,j}$ is furthermore the start position of task $O_{i,j+1}$. The tour $T_i = (q_{i,1}, \ldots, q_{i,2n_i+2})$ is the sequence of positions to be visited by R_i which starts at the depot, passes through the start and end positions of all tasks in the given order, and finally returns home to d_i. The processing time of task $O_{i,j}$ will be denoted by $p_{i,j}$. For simplicity we will write $p^w_{i,j}$, $p^d_{i,j}$ for the processing time of $W_{i,j}$ respective $D_{i,j}$. As usual in job-shop scheduling, we will assume all processing times to be integral. Let $P_i := \sum_{i=1}^{2n_i+1} p_{i,j}$ denote the sum over all welding and driving times of robot R_i.

The set C_{ll} of line-line collisions consists of pairs of tasks $(O_{i,j}, O_{i',j'})$ where simultaneously processing may lead to a collision. The set C_{lp} of line-point collisions consists of pairs $(O_{i,j}, q_{i',j'})$ where the processing of $O_{i,j}$ may lead to a collision with robot i' resisting at position $q_{i',j'}$. We assume $(O_{i,j}, Q_{i',j'}) \in C_{\mathrm{ll}}$ whenever $(O_{i,j}, q_{i',j'}) \in C_{\mathrm{lp}}$ or $(O_{i,j}, q_{i',j'+1}) \in C_{\mathrm{lp}}$. Note that line-line collisions are defined in a symmetric way, while (in general) the line-point collisions are not. Whenever laser source l switches between robots there is an integral delay of τ^l. The goal is to minimize the makespan, i. e. the time the last robot arrives back at its depot.

A *solution* to *LSP-T* is an assignment of laser sources to robots and of start times to tasks, such that

- robots assigned to identical laser source do not weld simultaneously
- all moves are collision free according to C_{ll} and C_{lp}

3 A polynomial algorithm for two robots

As already pointed out in the introduction for a given assignment of robots to laser sources, *LSP-T* without latency and any collisions corresponds to a job-shop problem. This two job job-shop problem can be formulated as a shortest path problem in the plane with rectangular obstacles. This main idea already appeared in Akers and Sheldon [2]. Brucker [3] showed that this is equivalent to a shortest path problem in an appropriate directed acyclic graph. As a consequence the two job job-shop problem was shown to be solvable in polynomial time.

In this section we will show how to extend the geometric approach of Brucker to *LSP-T* with two robots. We start with a brief description of the original algorithm of [3] which solves *LSP-T* with a single laser source without any collisions and latency. Then we show how collision avoidance can be incorporated and finally we integrate the treatment of positive latency. Note that whenever there are at least two laser sources available, every robot will get one on its own. Thus the laser source assignment is trivial for two robots.

3.1 The Geometric Algorithm of Brucker

Formulating the problem as a shortest path problem in the plane with obstacles works as follows: The x axis ranges from 0 to P_1 and the y axis from 0 to P_2. Each task $O_{i,j}$ will be associated with the interval $[a, b]$ with $a := \sum_{h=1}^{j-1} p_{i,h}$, $b := a + p_{i,j}$. Thus a is the earliest possible start time for task $O_{i,j}$. Note that for each R_i the intervals corresponding to tasks $O_{i,j}$ partition $[0, P_i]$.

The x and y coordinates of every point in $[0, P_1] \times [0, P_2]$ represent possible states of the robots: if $x \in (a, b)$ belonging to $O_{1,j}$ and $x = a + t$ then robot R_1 is currently processing $O_{i,j}$ and has still $p_{i,j} - t$ time units to proceed for

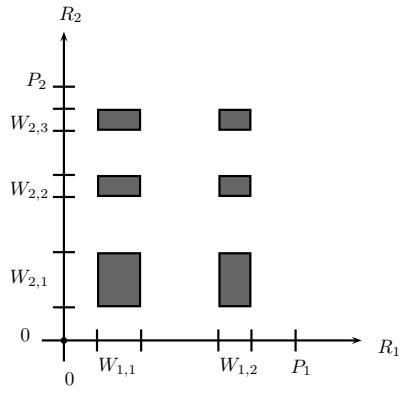

Figure 1: Plane with obstacles for two robots

finishing it. The y-coordinate is interpreted similarly. In job-shop for each pair of task $O_{1,j}, O_{2,j'}$ with $\mu_{1,j} = \mu_{2,j'}$ a rectangle is drawn (see figure 1) in order to forbid parallel processing.

A solution to this job-shop problem corresponds to a path in the plane from $(0, 0)$ to (P_1, P_2) which avoids the interior of all obstacles and uses only horizontal (R_1 working), vertical (R_2 working) or diagonal (both working) segments. Consequently, the *length* of a horizontal or vertical segment is the euclidean length, i. e. Δx respective Δy. For the diagonal parts it is defined as the euclidean length of the projection on the x axis (or equivalently on the y axis). For every such path from $(0, 0)$ to (P_1, P_2) the total length is the makespan of the associated schedule.

Brucker showed that this problem can be transformed into an ordinary shortest path problem in a directed acyclic graph $G = (V, A)$ which can then be solved in $\mathcal{O}(|V| + |A|)$ by dynamic programming. For this purpose he introduced one vertex for the point $(0, 0)$, one for (P_1, P_2) and two vertices for every obstacle, representing the north west and the south east corner. From every vertex v he started a line sweep moving diagonal upwards until he hits another obstacle or one of the border lines $x = P_1$ respective $y = P_2$. In the first case the vertex will be connected with the two vertices of the hit obstacle by two arcs (see figure 2), in the second one v will be connected with the vertex (P_1, P_2). The weight of the arcs is set to the length of the path from v to the corresponding corner.
Because every obstacle has 2 vertices and every vertex at most two arcs, the shortest path can be solved in $\mathcal{O}(n)$ where n is the number of obstacles, i. e. $n = n_1 \cdot n_2$ in the case of *LSP-T*. Brucker also showed the G can be constructed using $\mathcal{O}(n \cdot \log n)$ operations.

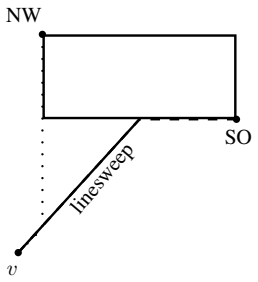

Figure 2: Construction of Arcs

3.2 Integrating Collision Avoidance

For handling line line and line point collisions the geometric algorithm needs three modifications: First, instead of drawing a rectangle for every pair of tasks on the same machine, we just demand that they share at least one common resource. Thus, we draw a rectangle between $O_{1,j}$ and $O_{2,j'}$ if both are welding jobs and there is only one laser source or if $(O_{1,j}, O_{2,j'}) \in C_{\text{ll}}$.

Second, let $(O_{i,j}, q_{i',j'}) \in C_{\text{lp}}$ where $q_{i',j'} \neq d_{i'}$, i.e. $R_{i'}$ is not allowed to wait at $q_{i',j'}$ while R_i is processing $O_{i,j}$. Remember: we have$(O_{i,j}, O_{i',j'}), (O_{i,j}, O_{i',j'+1}) \in C_{\text{ll}}$ and thus two neighboring rectangles. Taking care of these line point collision can therefore be simply done by unifying these two rectangles.

For line point collisions of the form $(O_{1,j}, q_{2,1})$ involving the depot of robot R_2 we modify the network constructed by Brucker's algorithm. We must force R_2 to leave d_2 before R_1 can start the processing of $O_{1,j}$. This means forbidding the horizontal line segment $O_{1,j}$ and can be done by removing the vertex associated with the south eastern corner of the obstacle defined by $O_{1,j}$ and $O_{2,1}$. In the case of $(O_{1,j}, q_{2,n_2+2}) \in C_{\text{lp}}$ the task $O_{1,j}$ has to be processed before O_{2,n_2+1}. So we stretch the rectangle on the left-hand side to the y-axis (see figure 3) and remove the vertex of the north west corner to forbid the vertical strip afterwards.

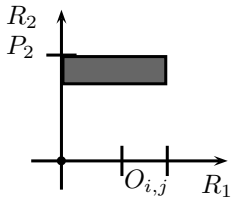

Figure 3: Line Point collision with depot

3.3 Handling the latency time

Now let $\tau^l > 0$: Tuchscherer et al. [5] observed for the case $C_{\text{lp}} = C_{\text{ll}} = \emptyset$ that *LSP-T* can be transformed into an equivalent problem with $\tau^l = 0$ if τ^l is sufficiently small, i. e. smaller then all driving times. Their modification increases the welding time of all jobs by τ^l and reduces the following driving time accordingly.

This transformation is not valid anymore when collisions are present: Consider the example in Figure 4. Here we are given two welding tasks and two driving tasks which have a line-line collision. In the transformed problem the no wait schedule is feasible while it induces a collision in the original one.

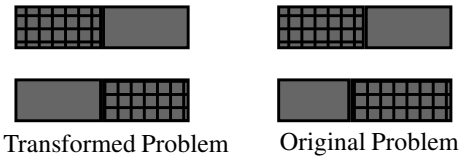

Figure 4: τ^l cannot be removed

Let $W_{1,j}, W_{2,j'}$ be two welding tasks. We have to take care that at least τ^l time units have past after the finishing of $W_{1,j}$ before the processing of $W_{2,j'}$ can start, or vice versa. In the following we will concentrate on the first situation the other is analogous.
If the next driving move $D_{1,j+1}$ does not induce a line-line collision with $W_{2,j'}$ we can "locally" increase the welding time of $W_{1,j}$ stretching the rectangle to the right. If $D_{1,j+1} \leq \tau^l$ we are done, otherwise we can only increase it by $D_{1,j+1}$. But now the two obstacles $(W_{1,j}, W_{2,j'})$ and $(W_{1,j+1}, W_{2,j'})$ are side-by-side. If we apply the linesweep algorithm we get a vertical arc from the south east corner of $(W_{1,j}, W_{2,j'})$ to the north west corner of $(W_{1,j+1}, W_{2,j'})$ with weight $p^w_{2,j'}$. This weight only needs to be increased by $\tau^l - p^d_{1,j+1}$ (see figure 5).
If $(D_{1,j+1}, W_{2,j'}) \in C_{\text{ll}}$ then we already have adjacent rectangles and thus their size does not need to be changed. The linesweep algorithm will produce two vertical arcs of weight $p^w_{2,j'}$: one from the south east corner of $(W_{1,j}, W_{2,j'})$ to the north west corner of $(D_{1,j+1}, W_{2,j'})$ and one from the south east corner of $(D_{1,j+1}, W_{2,j'})$ to the north west corner of $(W_{1,j+1}, W_{2,j'})$. The weight of the first arc will be increase by τ^l and the weight of the second one by $\max\{\tau^l - p^d_{1,j+1}, 0\}$. Note, if some of the rectangles have been unified by line-point collision handling before, then the weight changes of the non existent arcs will not be carried out.
We finally state the main result of this section:

Proposition 1. *LSP-T with $|R| = 2$ can be solved in $\mathcal{O}(n \log n)$ where n is the number task pairs sharing a common resource, i. e. a laser source or a line-line collision.*

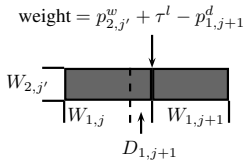

Figure 5: Integrating Latency

Proof. Define an obstacle for every task pair sharing a common resource with all modification subject to collisions and latency as described above. This requires $\mathcal{O}(n)$ operations. Apply Brucker's algorithm to construct the network. This works in $\mathcal{O}(n \log n)$ (see [3]) and creates $2n$ vertices and $\leq 4n$ arcs.
Adjust the weights of the arcs for handling latency and remove vertices as needed for the line point collisions. This can be done in $\mathcal{O}(n)$. Finally, solve a shortest path problem in the network. Applying topological sort and dynamic programming this requires $\mathcal{O}(n)$. □

4 NP-hardness of three robots

4.1 The $\mathcal{NP}$-hardness result

We will reduce the even-odd partition problem to an instance of *LSP-T* that is described by three robots and one laser source. Furthermore the instance we use for the reduction is without any collisions and latency. Therefore this instance can also be seen as a special instance of the job-shop scheduling problem with three jobs and four machines. However three of these machines are solely used by one robot/job each. Therefore all the tasks of a robot/job scheduled on his own machine are conflict free with respect to the other two robots/jobs. Only the forth machine (corresponding to the laser source) has tasks of all of the three jobs assigned. So this subproblem of job-shop has a very simple structure and the hardness result of Sotskov and Shakhlevich [14] for $(J3|3|C_{max})$ does not apply to the instances we use. Their result heavily relies on the fact that all three machines have to process tasks of all three jobs. However since the instances we use is a job-shop instances with makespan objective we can concentrate on active schedules when asking for optimal solutions (cf. Pinedo [9]).

Definition 1. **Even-odd partition**

GIVEN: A multiset $B := \{e_1, e_2, ..., e_{2n-1}, e_{2n}\}$ of $2n$ integers such that $e_{2i} < e_{2i-1}$ for all i, $1 \leq i \leq n$.

QUESTION: Is there a subset $B' \subset B$ such that

1. B' contains exactly one of $\{e_{2i-1}, e_{2i}\}$ for all i, $1 \leq i \leq n$, and
2. $\sum_{e_i \in B'} e_i = \sum_{e_i \in B \setminus B'} e_i$?

The $\mathcal{NP}$-hardness of this problem already appeared in Garey and Johnson [4].

Definition of the *LSP-T* instance

Let I be an instance of even-odd partition. The instance J of *LSP-T* consists of three robots R_1, R_2 and R_3. For each pair (e_{2i-1}, e_{2i}) of integers in I ($i \in \{1, \ldots, n\}$) we define a corresponding sequence of tasks T_j^i for each of the three robots j in the following way (T_j^i is called block with index i of robot j in the following):

R_j	tasks with processing times T_j^i						
1	W_{11}^i	D_{11}^i	W_{12}^i	D_{12}^i	W_{13}^i	D_{13}^i	W_{14}^i
	e_{2i-1}	H	e_{2i}	3 3/4$H + \delta_i$	δ_i	2 1/2H	$2H - e_{2i-1}$
2	W_{21}^i	D_{21}^i	W_{22}^i	W_{23}^i	D_{22}^i	W_{24}^i	D_{23}^i
	H	e_{2i-1}	$2H$	$2H$	$4H$	H	$2H - e_{2i-1}$
3	D_{31}	W_{31}^i	W_{32}^i	D_{32}^i			
	$6H + 2e_{2i-1}$	H	H	$4H - e_{2i-1} - \delta_i$			

In this setting W_{jk}^i denotes welding tasks, D_{jk}^i denotes driving tasks, $\delta_i = e_{2i-1} - e_{2i}$ and $H > 8E$, where $E = \sum_{i \in B} e_i$. The full sequence of tasks for R_j is then defined by the sequence of blocks $T_j^1, T_j^2 \ldots T_j^n$. The figures 6 and 7 are important types of schedules of block 1 for all three robots. Note that for the ease of readability we just skipped all welding and driving of length 0 that would actually occur in our instance whenever a task of some type follows a task of the same type.

The basic idea of our construction relies on the proof of Sotskov and Shakhlevich [14] for $(J3|3|C_{max})$: all the blocks of R_1 and R_2 are synchronized, i.e. in scheduling solutions that are relevant for our proof all blocks T_1^i and T_2^i end at the same point in time. R_1 and R_2 together correspond to the set B' of the even-odd partition. Moreover the duration of all tasks of T_1^i and T_2^i encodes whether e_{2i} or e_{2i-1} is is added to B'. R_3 on the other hand corresponds to $B \setminus B'$. Figure 6 characterizes the situation that e_1 is added to B' and e_2 to $B \setminus B'$. Figure 7 describes the other possibility. What we will do in our proof is showing that for active schedules of weight $12 \cdot n \cdot H + E/2$ just these two situations are relevant. However for blocks with index $i > 1$, T_1^i (resp. T_2^i) and T_j^3 will be little bit displaced with respect to each other. But with the above setting of H we will show that this displacement does not lead to infeasible schedules and so we

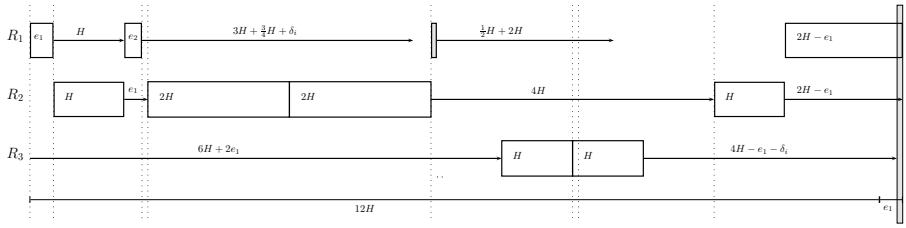

Figure 6: Block Structure $B11$

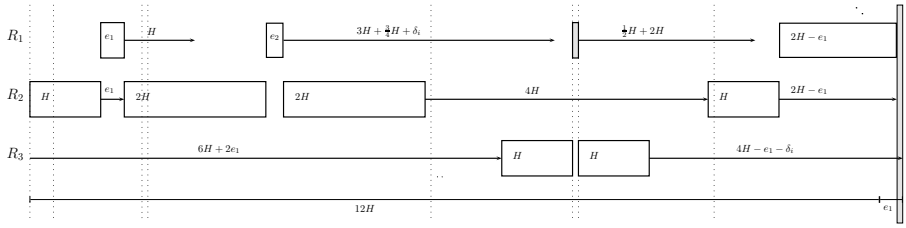

Figure 7: Block Structure $B21$

get the correspondence between even-odd partition and *LSP-T*. Note that even tough the basic idea relies on Sotskov and Shakhlevich [14], the construction we present differentiates in significant points from their approach, moreover the proof we present is simpler.

Scheduling the welding tasks of T_1^i, T_2^i and T_3^i as the sequence

$$(W_{11}^i, W_{21}^i, W_{12}^i, W_{22}^i, W_{23}^i, W_{13}^i, W_{31}^i, W_{32}^i, W_{24}^i, W_{14}^i)$$

on the laser source is called block structure B_{1i} (cf. figure 6). Scheduling them as sequence

$$(W_{21}^i, W_{11}^i, W_{22}^i, W_{12}^i, W_{23}^i, W_{31}^i, W_{13}^i, W_{32}^i, W_{24}^i, W_{14}^i)$$

is called B_{2i} (cf. figure 7). If all blocks T_1^i, T_2^i and T_3^i are scheduled as B_{1i} or B_{2i} and if they are finished before the welding tasks of T_1^{i+1}, T_2^{i+1} and T_3^{i+1} start, we call such a schedule *useful*.

As we will show for getting the connection between even-odd partition and *LSP-T* it suffices to consider only active schedules with $C_{max} = 12 \cdot n \cdot H + E/2$. On the other hand for instance J we can immediately get a sharp lower bound of $12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i}$ by looking at the minimal total time for all tasks of R_3 to be finished. By considering just the tasks of R_1 and R_2 and scheduling them according to B_{2i} for all i we get the same lower bound, moreover this bound is obviously sharp.

Lemma 1. *Scheduling all tasks of R_2 and R_3 in the trivial order can be done feasibly without causing any task to wait (not taking R_1 into account).*

Proof. We will consider the two robots separately: If we schedule all task of R_2 consecutively we get that W^i_{23} ends at $5H + e_{2i-1} + 12H(i-1)$. Task W^i_{24} starts exactly $4H$ time units later.
If we schedule all task of R_3 consecutively we get that W^i_{31} starts at $6H + 2e_{2i-1} + 12H(i-1) + \sum_{j=1}^{i-1} e_{2i}$ and task W^i_{32} ends exactly $2H$ time units later. Therefore the difference between the starting of W^i_{31} and the finishing of W^i_{23} can be upper bounded (resp. lower bounded) for all $i \leq n$ by the following expressions, where equality holds for $i = n$:

$$(H + e_{2i-1} + \sum_{j=1}^{n} e_{2i}) \leq H + E \quad (\text{resp. } H + e_{2i-1}) \tag{1}$$

The difference between the starting time of W^i_{24} and the finishing time of W^i_{32} can be upper bounded (resp. lower bounded) by:

$$H - e_{2i-1} \quad (\text{resp. } (H - e_{2i-1} - \sum_{j=1}^{n} e_{2i}) \geq 3/4H) \tag{2}$$

Again equality hold for $i = n$. So all tasks of R_2 and R_3 can be scheduled in the trivial order without any conflicts on the laser source, furthermore all tasks are processed without any waiting time in between.

□

By postponing the starting time of any of the tasks of R_2 by $E + 1$ (resp. $(\sum_{i=1}^{n} e_{2i-1} + 1 := E_o$ for R_3) time units a resulting schedule would have $C_{max} > 12 \cdot n \cdot H + E$. Therfore by the bounds (1) and (2) we get that the order of welding jobs of R_2 with respect to R_3 and vice versa has to be fixed by the trivial order for getting a schedule with $C_{max} \leq 12 \cdot n \cdot H + E$.

Lemma 2. *If all tasks are scheduled according to block structure B_{1i} or B_{2i} (for all i) a schedule that is feasible and active with $C_{max} \leq 12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i-1}$ can be found.*

Proof. $C_{max} \leq 12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i}$ is a lower bound due to R_3. If we use block structure B_{2i} for all i the upper bound due to robot R_3 increases by $\sum_{i=1}^{n} \delta_i$. This already gives the desired bound. With the help of bounds (1) and (2) of Lemma 1 the feasibility follows: if one uses only B_{2i} the difference between the starting time of W^i_{24} and the ending time of task W^i_{32} is not smaller than $(H - e_{2i-1} - \sum_{j=1}^{n} e_{2i-1})$; and still D^i_{13} ends before W^i_{24} does so.

On the other hand if one uses only B_{1i} the difference between the starting time of W^i_{31} and the ending time of task W^i_{23} is not smaller than $H + e_{2i-1} - \sum_{j=1}^{n} \delta_i$ and again D^i_{13} ends before W^i_{24} does so. Analyzing these extremal situations is sufficient, since a mixture of using block structure B_{1i} and B_{2i} leaves bigger gaps. □

Note that in this case the lower bound of $12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i}$ for R_3 is augmented by δ_i each time that B_{2i} is chosen. In the same way the lower bound of $12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i}$ for R_1 and R_2 is augmented by δ_i each time that B_{1i} is chosen.

Lemma 3. *In any active schedule of length smaller or equal to* $12 \cdot n \cdot H + E$ *the task* W_{14}^i *has to be performed directly after task* W_{24}^i.

Proof. It is not possible to schedule W_{14}^i between W_{21}^j and W_{22}^j or between W_{22}^j and W_{23}^j (for any j) since this would cause R_2 to wait longer than $E + 1$ time units. The same argument works for W_{31}^j and W_{32}^j of R_3. Because of the bounds (1) derived in Lemma 1 scheduling W_{14}^i between W_{23}^j and W_{31}^j would cause a violation of the bound on C_{max}. Scheduling it between W_{32}^j and W_{24}^j is impossible because of bounds (2). Scheduling W_{13}^i after W_{24}^j followed by W_{14}^i would also result in a violation of the bound, since W_{21}^{i+1} would be postponed by more than $E + 1$ units of time. So it follows that W_{14}^i is scheduled after W_{24}^i. It remains to show that this is done without any time lag in between. Assume that there is a time lag. Since the schedule is active this means that D_{13}^i finishes after W_{24}^i does so. This can only be caused by the fact that W_{12}^i is scheduled not before W_{23}^i. This implies that W_{21}^{i+1} cannot start earlier than 8 $1/4H$ time units after the finishing of W_{23}^i (by bounding the processing times of D_{12}^i, D_{13}^i and W_{14}^i). This is a contradiction to the time bound since W_{21}^{i+1} has to be postponed for more than $E + 1$ time units. Scheduling W_{13}^i after W_{32}^i gives again a contradiction by the time bound (2) of Lemma 1. □

With this we have that R_1 and R_2 are synchronized and the processing of tasks of R_2 and R_3 is fixed with respect to each other in schedules with $C_{max} \leq 12 \cdot n \cdot H + E$. Furthermore by Lemma 2 and Lemma 3 in such a schedule D_{31}^{i+1} starts directly after D_{32}^i has finished - the same is true for W_{21}^{i+1} and D_{23}^i.

Lemma 4. *In any active schedule of length smaller or equal to* $12 \cdot n \cdot H + E$ *the task* W_{13}^i *is scheduled exclusively between* W_{31}^i *and* W_{32}^i *or directly before* W_{31}^i.

Proof. Assume that any other welding job $\tilde{w}$ besides from W_{13}^i is scheduled between W_{31}^i and W_{32}^i. If $\tilde{w}$ has length greater or equal to H, W_{32}^i would be postponed for more than E_o time units. Task W_{12}^i remains a possible candidate for being scheduled in this position. Since the earliest possible starting time for welding task W_{14}^i is now $7H + 2\delta_i - e_{2i-1}$ time units after the ending time of W_{12}^i together with Lemma 3 we get a contradiction. All other tasks with length smaller than H can be ruled out by similar arguments.

Scheduling W_{13}^i after W_{32}^i was already ruled out in the proof of Lemma 3.

Assume that W_{13}^i is scheduled before W_{31}^i but not directly before it. Since the order of welding jobs of R_2 and R_3 is fixed with respect to each other we can assume w.l.o.g. that W_{13}^i is scheduled directly before W_{23}^i. But by taking the sum of the lengths of W_{11}^i, D_{11}^i, W_{12}^i, D_{12}^i and W_{13}^i together with Lemma 3 we get a delay of more than $E + 1$ time units for W_{23}^i, a contradiction.

□

Lemma 5. *If an active schedule S of length smaller or equal to $12 \cdot n \cdot H + E$ exists, we can find a useful schedule with total length not larger than S.*

Proof. By Lemma 4 we have to distinguish the following two cases:

Case 1. W^i_{13} is scheduled directly before W^i_{31}: if W^i_{11} is scheduled after W^i_{21} and directly followed by W^i_{12}, task W^i_{21} is postponed by more than $E+1$ units of time. If W^i_{12} is scheduled after W^i_{22}, W^i_{31} is postponed by more than $E_o + 1$ time units. Therefore S is *useful*.

Case 2. W^i_{13} is scheduled between W^i_{31} and W^i_{32}: if W^i_{11} is scheduled directly after W^i_{21} (resp. directly after W^i_{31}) and directly followed by W^i_{12} or if W^i_{12} is scheduled after W^i_{23} we again get a contradiction with respect to the time bound. So two possible non-*useful* sequences for S remain: $(W^i_{11}, W^i_{21}, W^i_{12}, W^i_{22}, W^i_{23}, \ldots)$ and $(W^i_{11}, W^i_{21}, W^i_{22}, W^i_{12}, W^i_{23}, \ldots)$. In both cases using the sequence of B_{2i} decreases the total processing time of the tasks of T^i_1 and T^i_2, in the first case by δ_i in the second case by e_{2i-1}. Furthermore this sequence remains feasible by Lemma 2. □

Lemma 6. *C_{max} is greater or equal to $12 \cdot n \cdot H + E/2$.*

Proof. Considering just the tasks of R_1 and R_2 and scheduling them according to B_{2i} for all i gives the sharp lower bound of $12nH + \sum_{i=1}^{n} e_{2i}$. Assume that there is a schedule giving a $C_{max} < 12 \cdot n \cdot H + E/2$. By Lemma 5 there exists also a *useful* schedule S with this property. However if the tasks of T^i_1 and T^i_2 in the *useful* schedule need a processing time of $12H + e_{2i}$ then the tasks of T^i_3 need a processing time of $12H + e_{2i} + \delta_i$ and vice versa. Therefore either all tasks of R_1 and R_2 together need a total processing time not less than

$$12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i} + 1/2 \sum_{i=1}^{n} \delta_i$$

or alls tasks of R_3 together need a total processing time not less than this bound. This contradicts that $C_{max} < 12 \cdot n \cdot H + E/2$.

□

Theorem 1. *LSP-T is already $\mathcal{NP}$-hard for three robots, one laser source and without any collisions and latency.*

Proof. $\Longrightarrow$ Assume that the even-odd partition I has a solution such that $\sum_{e_j \in B'} e_j = \sum_{e_j \in B \setminus B'} e_j$. Construct J in the following way: if $e_{2i} \in B'$ use B_{2i} otherwise use B_{1i}. By simple algebra it follows that $C_{max} = 12 \cdot n \cdot H + E/2$

$\Longleftarrow$ Let a solution of J be given such that $C_{max} = 12 \cdot n \cdot H + E/2$. By Lemma 5 and Lemma 6 we can find a *useful* schedule S with the same makespan. If B_{2i} was used we construct an even-odd partition instance by adding e_{2i} to B' otherwise we add e_{2i-1} to B'. Assume now that $\sum_{e_j \in B'} e_j \neq \sum_{e_j \in B \setminus B'} e_j$. W.l.o.g let $\sum_{e_j \in B'} e_j < \sum_{e_j \in B \setminus B'} e_j$. Since S is active we get that the total completion time of robots R_1 and R_2 is smaller than $12 \cdot n \cdot H + E/2$ (in fact

it is equal to $12 \cdot n \cdot H + \sum_{e_j \in B'} e_j$). By the same argument as in the proof of Lemma 6 we get that the sum of completion times of robots R_1 and R_2 on the one hand and of robot R_3 equals:

$$24 \cdot n \cdot H + \sum_{i=1}^{n} (e_{2i-1} + e_{2i}) = 24 \cdot n \cdot H + E \qquad \lightning$$

□

5 A pseudo polynomial algorithm and an FPTAS

In this section we will show that *LSP-T* for a fixed number of robots can be solved in pseudo polynomial time. By complete enumeration we can also assume w. l. o. g. that the assignments of laser sources to robots is fixed.

5.1 The pseudo polynomial algorithm

Our algorithm uses a transversal graph approach as introduced to scheduling problems by Middendorf and Timkovsky [7]. We will construct a directed acyclic graph $G = (V, A)$ with unit weigh arcs where a shortest path coincides with an optimal solution to *LSP-T*.

For $i = 1, \ldots, k$ let $O_{i,0}$ with $p_{i,0}$ be an artificial start task. Our vertex set V consists of a terminal vertex T together with tuples $(i_1, t_1, \ldots, i_k, t_k, j_1, d_1, \ldots, j_h, d_h)$ with the following meaning:

- $i_r \in \{0, \ldots, 2n_r + 1\}$ is the index of the task currently being processed by robot R_r.
- $t_r \in \{0, \ldots, p_{r,i_r}\}$ denote how long R_r has already worked on O_{r,i_r}.
- $j_l \in \{-1, 1, \ldots, k\}$ is the index of the robot who used laser source L_l last. Here $j_l = -1$ means, that the laser source has not be used so far.
- $d_l \in \{0, \ldots, \tau^l\}$ is the time laser source L_l has been idle since its last usage.

Let $v = (i_1, t_1, \ldots, i_k, t_k, j_1, d_1, \ldots, j_h, d_h) \neq T$ be a vertex. Denote by

$$I_c := \{r \mid t_r < p_{r,i_r}\}$$

the set of robots currently working and by

$$\begin{aligned} I_n := \{r \mid\ & i_r < 2n_r + 1, t_r = p_{r,i_r} \\ & \text{and if } O_{r,i_r+1} \text{ is a welding task additional} \\ & j_{l(R_r)} = r \vee j_{l(R_r)} = -1 \vee d_{l(R_r)} = \tau^{l(R_r)}\} \end{aligned}$$

the set of robots which can start their next task. Here $l(R_r)$ denotes the index of the laser source robot R_r is attached to.

Denote by $\mathcal{I}$ the set of all subsets of $I_c \cup I_n$ which can work in parallel. This means, I induces no line-line collision, includes at most one welding task per laser source and induces no line-point collisions with any robot $r \notin I$. For every $I \in \mathcal{I}$ we create an arc of weight one from v to $w = (i'_1, t'_1, \ldots, t'_k, i'_k, j'_1, d'_1, \ldots, j'_h, d'_h)$ defined by

- $i'_r = i_r + 1, r \in I \cap I_n$, $i'_r = i_r$ otherwise
- $t'_r = t_r + 1, r \in I \cap I_c$, $t'_r = 1$, $r \in I \cap I_n$, $t'_r = t_r$ otherwise
- $j'_l = r$ if $r \in I \cap I_n$ with R_r starts a welding task on L_l, $j'_l = j_l$ otherwise
- $d'_l = 0$ if $r \in I \cap I_N$ with R_r starts a welding task on L_l, $d'_l = d_l + 1$ if no robot in I with $l^r = l$ processes a welding task

So every robot in I processes a task for one time unit. Every vertex with $i_r = n_r$ for all robots will be connected with the terminal vertex T by an arc of zero weight.

Observation 1. *Every path in G from $(0, \ldots, 0)$ to T corresponds to a feasible schedule of LSP-T where the makespan is the sum over the arcs weights of P. Moreover every non preemptive active schedule can be written in this form.*

Given such a path P we can extract the start times of the k-th task of robot r by summing up the arc weights of P between vertices with $i_r < k$.

Observation 2. *The number of vertices in G can be bounded by*

$$|V| \leq \max_{r=1}^{k}(2n_r + 2)^k \cdot \max_{r=1}^{k} \max_{j=1}^{n_r} p_{r,j}^k \cdot \max_{l=1}^{h}(\tau^l)^h$$

and there are at most $2^k|V|$ arcs. Because $h \leq k$ and k is constant, the size of G is pseudo polynomial in the size of LSP-T.

Theorem 2. *LSP-T can be solved in pseudo-polynomial time when the number of robots is constant.* □.

In section 4 we proved that *LSP-T* with three robots, no latency and no collisions is already $\mathcal{NP}$-hard. However, there is a polynomial solveable special case.

Proposition 2. *Let $C_{ll} = C_{lp} = \emptyset$ and $\tau^l = 0, l = 1, \ldots, h$. If*

$$\max_{i=1}^{k} \max_{j=1}^{n_i} p_{i,j}^d \leq \min_{i=1}^{k} \min_{j=1}^{n_i} p_{i,j}^w \tag{3}$$

then LSP-T can be solved in polynomial time.

Proof. In the absense of collisions *LSP-T* decomposes into h problems with a single laser source. We can therefore assume $h = 1$. We will construct a transversal graph similar to the above but it does not contain any time index in the vertices. V consists of a source vertex S, a terminal vertex T and vertices of the form $v = (i_1, \ldots, i_k, j)$ with $i_r \in \{0, \ldots, n_r\}, r = 1, \ldots, k$ and $j \in \{1, \ldots, k\}$. Here i_r is the index of the last performed welding task of robot R_r and j is the index of the robot worked last.
Let $v = (i_1, \ldots, i_k, j)$ be a vertex. For every r with $i_r < n_r$ we add an arc to the node $w = (i_1, \ldots, i_{r-1}, i_r + 1, i_{r+1}, \ldots, i_k, r)$. The weight of this arc will be set as follows: If $r = j$ then R_r has to move to its next welding task and thus the weight is $p^d_{r,i_r+1} + p^w_{r,i_r+1}$. Otherweise by (3) robot R_r has already done the move D_{r,i_r+1} while R_j welded. Thus the arc weight is only p^w_{r,i_r+1}.
For every r we connect S and $w = (i_1, \ldots, i_k, r)$ with $i_r = 1$, $i_{r'} = 0$, $r' \neq r$ by an arc of weight $p^d_{r,1} + p^w_{r,1}$. Every vertex $v = (n_1, \ldots, n_k, j)$ has an arc of weight p^d_{j,n_j+1} to the terminal vertex T.
Now it is easy to see that a shortest path from S to T in the directed acycled graph G yields an optimal solution to *LSP-T*. Futhermore the size of G is polynomial in the input size of *LSP-T*. □

5.2 A fully polynomial approximation scheme for *LSP-T*

We close this section by providing an FPTAS for *LSP-T* with a constant number of robots:

Theorem 3. *There is a fully polynomial approximation scheme (FPTAS) for LSP-T.*

Proof. We will use standard rounding argument which has been introduce by Ibarra and Kim [6] for the knapsack problem and is widely used in scheduling (see e. g. Agnetis et al. [1]).

Let $\Delta := \max\{\max_{l=1}^{h} \tau^l, \max_{r=1}^{k} \max_{j=1}^{n_r} p_{r,j}\}$ be the maximum time distance and $N := \sum_{r=1}^{k} 2n_r + 1$, the total number of tasks. Define $Z := \epsilon \frac{\Delta}{N}$.
We construct a modified instance of *LSP-T* with $\tilde{\tau}^l := \lceil \frac{\tau^l}{Z} \rceil$ and $\tilde{p}_{r,j} := \lceil \frac{p_{r,j}}{Z} \rceil$. By construction we have $\tilde{\tau}^l, \tilde{p}_{r,j} \leq 1 + N\frac{1}{\epsilon}$. Thus all time distances of the modified problem are polynomially bounded in N and $\frac{1}{\epsilon}$. Therefore the size of the transversal graph described above is also polynomial bounded in N and $\frac{1}{\epsilon}$. Solving the shortest path problem yields a solution $\tilde{S}$ with makespan $T(\tilde{S})$. If we multiply all task start times by Z we obtain a feasible solution S of the original problem with makespan $T(S) = Z \cdot T(\tilde{S})$.

Now let S^* be an optimal solution for the original problem with makespan $T(S^*)$. We have to show $T(S) \leq (1 + \epsilon)T(S^*)$. To see this, transform S^* to the modified problem by forming an active schedule $\tilde{S}$ keeping the order of the tasks. Because rounding up enlarges the processing time of each task by at most

1, we obtain $T(\tilde{S}^*) \leq \frac{T(S^*)}{Z} + N$. Thus we have

$$\begin{aligned} T(S) &= Z \cdot T(\tilde{S}) \\ &\leq Z \cdot T(\tilde{S}^*) \\ &\leq T(S^*) + Z \cdot N \\ &= T(S^*) + \epsilon \underbrace{\Delta}_{\leq T(S^*)} \\ &\leq (1+\epsilon) T(S^*) \end{aligned}$$

□

6 Conclusion

In this article we fully settled the approximability status of *LSP-T* with a constant number of robots by deriving the exact boundary between polynomial time solvable and weakly $\mathcal{NP}$-hard together wit an FPTAS. This situation is relevant for the industrial application of our problem since practical instances are characterized by the fact that the number of robots involved is small (≤ 5).

From a theoretical point of view the $\mathcal{NP}$-hardness of Section 4 refines the boundary between polynomial time solvable and $\mathcal{NP}$-hard for job-shop scheduling by considering a a very special subproblem of job-shop scheduling.

Still interesting questions concerning *LSP-T* remain: are there special graph classes in the conflict graph representation of the problem with a non-constant number of robots that guarantee a similar approximability (resp. inapproximability) behavior as in standard job-shop scheduling.

References

[1] Alessandro Agnetis, Marta Flamini, Gaia Nicosia, and Andrea Pacifici. A job-shop problem with one additional resource type. *Journal of Scheduling*, pages 1–13, 2010. 10.1007/s10951-010-0162-4.

[2] Sheldon B. Akers Jr. A graphical approach to production scheduling problems. *Operations Research*, 4(2):244–245, 1956.

[3] P. Brucker. An efficient algorithm for the job-shop problem with two jobs. *Computing*, 40(4):353–359, 1988.

[4] M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to NP-completeness*. WH Freeman and Company, San Francisco, 1979.

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

[6] Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22:463–468, October 1975.

[7] Martin Middendorf and Vadim G. Timkovsky. Transversal graphs for partially ordered sets: Sequencing, merging and scheduling problems. *Journal of Combinatorial Optimization*, 3:417–435, 1999.

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

[9] M. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer Verlag, 2008.

[10] 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.

[11] J. Rambau and C. Schwarz. Exploiting combinatorial relaxations to solve a routing & scheduling problem in car body manufacturing. Preprint, Universität Bayreuth, 2010.

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

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

[14] Y.N. Sotskov and NV Shakhlevich. NP-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, 59(3):237–266, 1995.

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

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