Noname manuscript No.
(will be inserted by the editor)

Job-Shop Scheduling in a Body Shop

Joachim Schauer · Cornelius Schwarz

Abstract We study a generalized job-shop problem called the *body shop scheduling problem* (*BSSP*). This problem arises from the industrial application of welding in a car body production line, where possible collisions between industrial robots have to be taken into account. *BSSP* corresponds to a job-shop problem where the operations of a job have to follow alternating routes on the machines, certain operations of different jobs are not allowed to be processed at the same time and after processing an operation of a certain job a machine might be unavailable for a given time for operations of other jobs. As main results we will show that for three jobs and four machines the special case where only one machine is used by more than one job is already $\mathcal{NP}$-hard. This also implies that the single machine scheduling problem that asks for a makespan minimal schedule of three chains of operations with delays between the operations of a chain is $\mathcal{NP}$-hard. On the positive side, we present a polynomial algorithm for the two job case and a pseudo-polynomial algorithm together with an FPTAS for an arbitrary but constant number of jobs. Hence for a constant number of jobs we fully settle the complexity status of the problem.

Keywords job-shop, FPTAS, complexity, transversal graph

1 Introduction

The real world application which motivated this article arises from welding on car bodies: Let $J = \{J_1, \ldots, J_k\}$ be a set of arc welding *robots*. Every robot J_i has to perform n_i *welding operations* $W_{i,j}$ ($j \in \{1, \ldots, n_i\}$). Each robot is supplied with energy by being physically connected to one out of h *laser sources*. This means that the assignment of robots to laser sources has to be done before the schedule starts and cannot be changed later. Each laser source can only serve one robot at a time, i.e. welding operations of robots assigned to identical laser sources cannot be processed in parallel. After finishing its current welding operation each robot has to perform a transversal move, called *driving operation*, $D_{i,j}$ to reach the start position of its next welding operation. Thus the operations of robot J_i are of the following form:

$$(O_{i,1}, \ldots, O_{i,2n_i+1}) = (D_{i,1}, W_{i,1}, D_{i,2}, \ldots, W_{i,n_i}, D_{i,n_i+1})$$

Here the first and the last driving operation model the fact that a robot has to start and end its tour at a given depot. We will focus on the makespan objective $C_{\max}$ which corresponds to the completion time of the last robot. Note that $h \leq k$ since otherwise a laser source would be idle.

In the real world application the number of robots is small, therefore we study the problem with the additional restriction that the number of robots is bounded by a constant. Note that the general problem was studied by Rambau and Schwarz (2010a) and Rambau and Schwarz (2010b): they presented a branch and bound algorithm, however they did not treat the approximability.

So let k be bounded by a constant. In this case complete enumeration yields a constant number of possible assignments of robots to laser sources. Note that all the algorithms presented in this paper are based on a fixed assignment of robots to laser sources. Therefore from a

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

Chair of Economics Mathematics, University of Bayreuth, D-95440 Bayreuth, Germany, E-mail: cornelius.schwarz@uni-bayreuth.de

computational complexity perspective one can perform these algorithms h^k times for every possible assignment of robots to laser sources and take the best solution over all runs. Hence we can w.l.o.g. assume from now on that the best assignment is known in advance. We will call the presented problem the *body shop scheduling problem* (*BSSP*).

So far *BSSP* could be modeled as a classical *Job-Shop Scheduling Problem* which is usually described in the following way: there are m *machines* $M_1, \ldots, M_m$ and k *jobs* $J_1, \ldots, J_k$ given. Each job J_i consists of n_i *operations* $O_{i,j}$ ($j \in 1, \ldots, n_i$). Every operation $O_{i,j}$ has to be performed $p_{i,j}$ time units on a dedicated machine $\mu_{i,j}$. In addition the order of the operations of every job is prescribed, i.e. we have precedence constraints of the form of chains. $O_{i,j} \rightarrow O_{i,j+1}$ means that operation $O_{i,j}$ has to be finished before $O_{i,j+1}$ can start (cf. e.g. Pinedo (2008)). Note that the jobs of job-shop correspond to the robots of the industrial application.

However in *BSSP* two additional parts play a crucial role. First, whenever a laser source l performs two operations 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 (2010b): Let q_i respective $q_{i'}$ be two positions such that robot J_i at q_i will collide with robot $J_{i'}$ at $q_{i'}$. If J_i bypasses q_i processing operation $O_{i,j}$ and $J_{i'}$ bypasses $q_{i'}$ processing $O_{i',j'}$ then we do not allow the pair $(O_{i,j}, O_{i',j'})$ of operations to be processed at the same time. We call such a pair *line-line collision*.

However considering only moving robots is not enough to guarantee the absence of collisions. If robot J_i processing operation $O_{i,j}$ collides with robot $J_{i'}$ resting at the end position of operation $O_{i',j'}$ then this is called a *line-point collision*. We will consider the end and start positions of operations as part of it, therefore a line-point collisions also implies two line-line collisions which are $(O_{i,j}, O_{i',j'})$ and $(O_{i,j}, O_{i',j'+1})$.

1.1 *BSSP* generalizes Job-Shop Scheduling

Every job-shop instance I can be written as a *BSSP* instance J in the following way: Every job corresponds to a robot and gets assigned a unique laser source on its own. Every operation of I is mapped to a welding operation of J with driving operations of zero length in-between. For every pair of operations processed on the same machine in I a line-line collision is introduced in J. This means that the machines of I can be modeled as collisions in J. From a more general perspective *BSSP* is a generalization of job-shop scheduling since in addition to the latency time on the laser sources it introduces three different types of constraints on the operations (laser sources, line-line collisions and line-point collisions) whereas job-shop scheduling introduces only one type of constraint (machines). In the following the three different types of constraints of *BSSP* will be denoted as *resources*.

From now on we will use the standard job-shop notation for *BSSP*. This means that we will speak of jobs instead of robots. A laser source can be modeled as a classical job-shop machine and will be called *general machine*. Furthermore one can model the driving operations $D_{i,j}$ of each job by introducing a job-shop machine for each job i which only performs the driving operations of i. We will denote these type of machines as *restricted machines*. Analogously we will call the welding operations *general operations* and the driving operations *restricted operations*.

1.2 Main Contributions

We will fully settle the complexity status of *BSSP* with a constant number of jobs. We will show that the problem is already $\mathcal{NP}$-hard for three jobs, one general machine and three restricted machines without any collisions and latency. Note that due to the absence of latency and collisions this *BSSP* subproblem is a job-shop subproblem and the hardness result is also interesting from a job-shop perspective since it complements the hardness result of Sotskov and Shakhlevich (1995) for $(J3|n = 3|C_{max})$. The subproblem we use is more symmetric than the one used by Sotskov and Shakhlevich (1995) in the sense that three of the four machines only perform operations of one dedicated job. Note that the hardness proof of Sotskov and Shakhlevich (1995) for $(J3|n = 3|C_{max})$ heavily relies on the fact that the three jobs perform operations on all of the three machines. As a consequence the application of their proof to our subproblem is not possible - the alternating chain structure of general and restricted operations cannot be incorporated by their setting.

On the positive side we will provide a polynomial time algorithm for two jobs based on the geometric approach to job-shop scheduling due to Akers (1956) and Brucker (1988). This provides a first sharp boundary. Moreover for a constant number of jobs we will give a pseudo-polynomial time algorithm by using a transversal graph approach introduced to scheduling theory by Middendorf and Timkovsky (1999). This algorithm will be used

for deriving an FPTAS by applying a standard rounding argument. By using a strong $\mathcal{NP}$-hardness result of Wikum et al. (1994) that applies to *BSSP* instances where the number of robots is part of the input, we get a second sharp boundary since in that case no FPTAS can exist (unless $\mathcal{P} = \mathcal{NP}$).

1.3 Important subproblems of *BSSP*

$(1 \mid \text{chains}(l_{i,j}) \mid C_{\max})$ defines the special one machine scheduling, where the operations have to be processed according to precedence constraints that are described by chains. Whenever operation i is followed by operation j in a chain, $l_{i,j}$ time units have to pass between the processing of i and j (see e.g. Wikum et al. (1994)). $(1 \mid \text{chains}(l_{i,j}) \mid C_{\max})$ can be seen as a restricted subproblem of *BSSP*: each chain corresponds to a job, $l_{i,j}$ is modeled by the restricted operations, the operations of a chain correspond to general operations and one general machine is available. Note that Wikum et al. (1994) showed that this problem is strongly $\mathcal{NP}$-hard if the number of chains is part of the input and even if the number of operations per chain is bounded. This result is important for us since under standard assumptions it rules out the existence of an FPTAS (cf. Ausiello et al. (1999)) for *BSSP* instances where the number of jobs is part of the input.

Note that the subproblem where all processing times as well as all time lags are identical $((1 \mid \text{chains}(l), p_j = p \mid C_{\max}))$ becomes polynomial time solvable. The same is true for $(1 \mid \text{chains}(l_{i,j}) \leq p_k \mid C_{\max})$ where p_k is the minimum over all processing times (cf. Munier and Sourd (2003)). In his PhD thesis Wikum (1992) also proposed a pseudo-polynomial algorithm for the case where only two chains are present. This case is indeed polynomial time solvable by Section 3. Note that the main results of this paper carry over to generalized versions of $(1 \mid \text{chains}(l_{i,j}) \mid C_{\max})$.

1.4 Related Literature

A related laser sharing problem without collision avoidance has been introduced by Grötschel et al. (2006) by providing a mixed integer formulation. Schneider (2006) integrated the laser source assignment and the tour planning for the robots. A branch-and-bound algorithm for an extended variant of the problem can be found in Rambau and Schwarz (2009). Collision handling was added in Rambau and Schwarz (2010b). A pseudo-polynomial time algorithm for *BSSP* with a fixed number of collisions can be found in Rambau and Schwarz (2010a). A related problem is the *welding cell problem (WCP)* studied by Welz (2010) and by Welz and Skutella (2011). Here, the task is to find collision free tours of point welding robots. The difference to *LSP* is that point welding robots do not share laser sources and the goal is to minimize operational costs instead of the makespan.

Section 2 introduces the notation of *BSSP*. In Section 3 we will describe the polynomial algorithm for the two job case. Section 4 covers the proof of the $\mathcal{NP}$-hardness for three jobs, 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 *BSSP* consists of k jobs $J_1, \ldots, J_k$, h general machines $M_1, \ldots, M_h$, where each job has to perform n_i general operations $W_{i,j}$ ($i = 1, \ldots, k$ and $j = 1, \ldots, n_i$) and $n_i + 1$ restricted operations $D_{i,j}$ ($i = 1, \ldots, k$ and $j = 1, \ldots, n_i+1$). The restricted operations are performed on k restricted machines. Furthermore sets C_{ll} of line-line collisions and C_{lp} of line-point collisions are given. Whenever the general machine l switches between operations of different jobs an integral delay of τ^l time units has to pass.

Each job J_i has associated a depot A_i. Each general- as well as restricted operation is characterized by a start position and an end position. The end position of operation $O_{i,j}$ is furthermore the start position of operation $O_{i,j+1}$. Note that using the term $O_{i,j}$ instead of the terms $W_{i,j}$ and $D_{i,j}$ is advantageous for properties that are common to restricted and general operations. The tour $T_i = (q_{i,1}, \ldots, q_{i,2n_i+2})$ is the sequence of positions to be visited by J_i. This sequence starts at the depot A_i, passes through the start and end positions of all operations in the given order, and finally ends at the depot A_i. The processing time of operation $O_{i,j}$ will be denoted by $p_{i,j}$. For improving the readability we will write $p^w_{i,j}$, $p^d_{i,j}$ for the processing times of $W_{i,j}$ and $D_{i,j}$. 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 general- and restricted operation times of job J_i.

The set C_{ll} of line-line collisions consists of pairs of operations $(O_{i,j}, O_{i',j'})$ where simultaneous 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 job i' residing at position $q_{i',j'}$. We assume $(O_{i,j}, O_{i',j'}) \in C_{\text{ll}}$ whenever

$(O_{i,j}, q_{i',j'}) \in C_{\text{lp}}$ or $(O_{i,j}, q_{i',j'+1}) \in C_{\text{lp}}$. Note that line-line collisions are defined in a symmetric way, while (in general) the line-point collisions are not. Let $p_{\max}$ be the maximal processing time over all operations, $\tau_{\max}$ be the maximal latency, and $n_{\max}$ the maximal number of operatins of any job. Then the length of an *BSSP* instance can be bounded by:

$$k \cdot n_{\max} * \lceil \log_2 p_{\max} \rceil + h \cdot \lceil \log_2 \tau_{\max} \rceil + |C_{ll}| + |C_{lp}|$$

Note that both $|C_{ll}|$ and $|C_{lp}|$ is $O((k \cdot n_{\max})^2)$. The goal is to minimize the makespan, i. e. the time the last job arrives back at its depot.

A *solution* to *BSSP* consists of assigning starting times to all the operations on their dedicated machines, such that

- the operations of a general machine are not processed simultaneously
- all operations are processed without collisions with respect to C_{ll} and C_{lp}
- the latency time τ_i of each general machine i is respected

Note that similar to job-shop scheduling when dealing with optimal solutions we can concentrate on *active* schedules. Active means that it is not possible to construct another schedule, through changes of the order of operations, where at least one operation is completed earlier but no other operation is completed later (cf. Pinedo (2008), Definition 2.3.3).

Lemma 1 *If an optimal schedule exists then there exists also an optimal schedule that is active.*

Proof Starting from a non-active optimal schedule S we can construct an active optimal one $\hat{S}$ in the following way. Since the input data is integer we can assume w.l.o.g. that all starting times of S are integer.
As long as the schedule is not active there exists a operation O which can be scheduled earlier without delaying any other operation. Pick this operation and schedule it as early as possible, i.e. such that the resulting schedule stays feasible. Since all starting times of S are integer this is also true for the completion times of all other operations. Thus, the starting time O will be decreased by at least 1 to another integer value. Because the makespan objective is regular, i.e. non decreasing in the completion times, the objective value of the modified schedule cannot be worse. In each iteration the starting time of a operation is decreased by at least one. Since it cannot be decreased below zero, this precedure terminates in a finite number of steps and we end up with an active schedule $\hat{S}$. □

3 A polynomial algorithm for two jobs

As already pointed out, for a given assignment of jobs to general machines and without latency and collisions *BSSP* corresponds to a job-shop problem. Following Akers (1956), we can formulate a job-shop problem with two jobs as a shortest-path problem in the plane with rectangular obstacles. Brucker (1988) showed that this is equivalent to a shortest-path problem in an appropriate directed acyclic graph. As a consequence, job-shop problems with two jobs were shown to be solvable in polynomial time.
In this section, we will show how to extend the geometric approach of Brucker to *BSSP* with two jobs. We start with a brief description of the original algorithm of Brucker (1988) which solves *BSSP* with a single general machine without any collisions and latency. Then we show how collision avoidance can be incorporated. Finally, we integrate the treatment of positive latency.

3.1 The geometric algorithm of Akers - Brucker's approach

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 operation $O_{i,j}$ will be associated with an interval $[a, b]$ where $a := \sum_{h=1}^{j-1} p_{i,h}$, and $b := a + p_{i,j}$. Thus, a is the earliest possible start time for operation $O_{i,j}$. Note that for each job J_i these intervals form a partition of $[0, P_i]$. The x- and y-coordinates of every point in $[0, P_1] \times [0, P_2]$ represent possible states of the jobs: if $x \in (a, b)$ belonging to $O_{1,j}$ and $x = a + t$ then job J_1 is currently processing $O_{i,j}$ and has still $p_{i,j} - t$ time units to proceed for finishing it. The y-coordinate is interpreted similarly. In job-shop for each pair of operation $O_{1,j}, O_{2,j'}$ 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 (J_1 working), vertical (J_2 working) or diagonal (both working) segments. Consequently, the *length* of a horizontal or vertical segment is the Euclidean length, i.e. Δx respectively 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

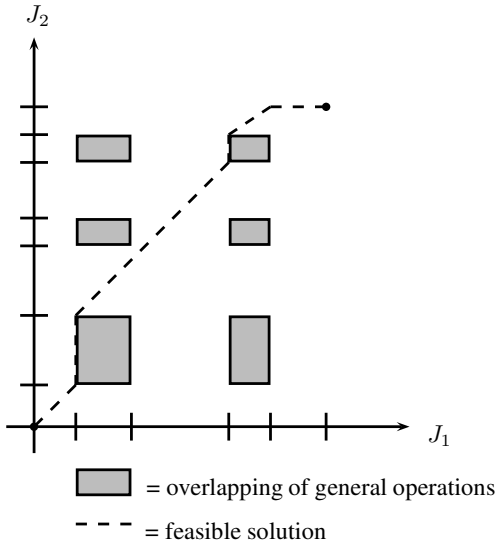
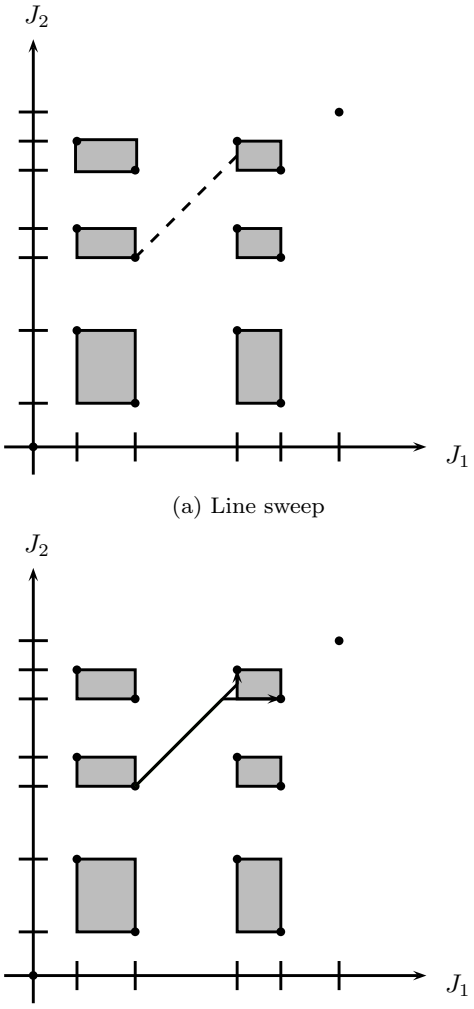


Fig. 1: Plane with obstacles for two jobs

$\mathcal{O}(|V| + |A|)$ 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 starts a line sweep moving diagonally upwards until he hits another obstacle or one of the border lines $x = P_1$ or $y = P_2$. 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.
Since every obstacle has two vertices and every vertex at most two arcs, the shortest-path problem in the acyclic graph G 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 *BSSP*. Brucker also showed that G can be constructed by using $\mathcal{O}(n \cdot \log n)$ operations.

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 operations on the same machine, we do so if they share at least one common resource. Thus, we draw a rectangle between $O_{1,j}$ and $O_{2,j'}$ if both are general operations and there is only one general machine available or if $(O_{1,j}, O_{2,j'}) \in C_{\text{ll}}$ (see Figure 3).
Second, let $(O_{i,j}, q_{i',j'}) \in C_{\text{lp}}$ where $q_{i',j'} \neq A_{i'}$, i.e. $J_{i'}$ is not allowed to wait at $q_{i',j'}$ while J_i is processing $O_{i,j}$. Recall: we have $(O_{i,j}, O_{i',j'}), (O_{i,j}, O_{i',j'+1}) \in C_{\text{ll}}$



(a) Line sweep

(b) Resulting arcs

Fig. 2: Construction of arcs

and thus two neighboring rectangles. Taking care of this line-point collision can therefore be done by unifying these two rectangles (see Figure 4).

Now we have to modify the line sweep algorithm because it may be that the line sweep hits a rectangle where the north-west corner is on the left side of the current vertex which means that J_2 has to travel into the past (see Figure 5). However, this can only occur if the line sweep starts at the blocking point of the line-point collision, e.g. in Figure 5 J_1 is currently blocking J_2, i.e J_1 is the one that resides at the blocking position. Thus, there is only one way to resolve this conflict: the blocking job (in our case J_1) has to move which means that the arc leading to the past, i.e. to the north west corner in the figure, will not be created.
The last modification covers line-point collisions of the form $(O_{1,j}, q_{2,1})$ involving the depot of job J_2. We have

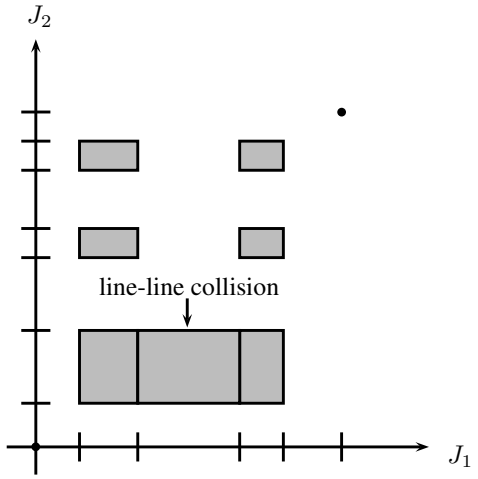
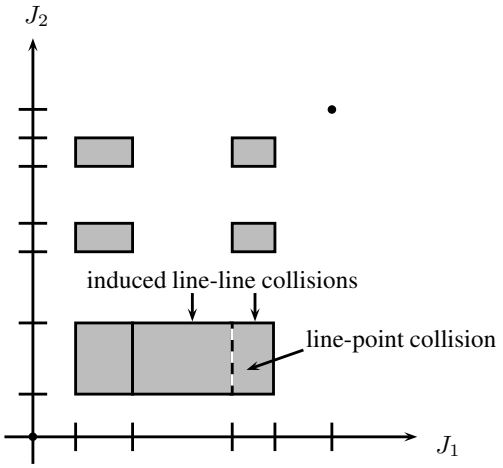


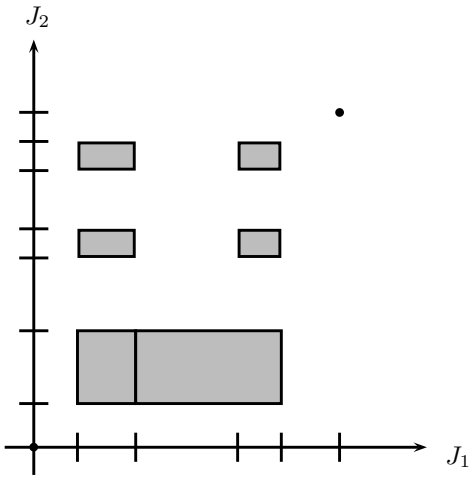
Fig. 3: Handling of line-line collision

to modify the network constructed by Brucker's algorithm. We must force J_2 to leave A_2 before J_1 can start the processing of $O_{1,j}$. This means forbidding the horizontal line segment $O_{1,j}$ and this can be done by removing the vertex associated with the south-east corner of the obstacle defined by $O_{1,j}$ and $O_{2,1}$. In this case also $(O_{1,j}, q_{2,n_2+2}) \in C_{\text{lp}}$. Therefore J_1 has to finish $(O_{1,j})$ before J_2 arrives back at A_2, in other words the operation $O_{1,j}$ has to be processed before O_{2,n_2+1}. This is accomplished by stretching the rectangle on the left-hand side to the y-axis (see Figure 6) and remove the vertex of the north-west corner to forbid the vertical strip afterwards.

Remark 1 To construct the network, Brucker's algorithm performs a line-sweep with the line $y - x = c$. This line is moved parallel from north-west to south-east. For each c the algorithm keeps a list S of intersected obstacles ordered lexicographically according to their north-west corners. Whenever the line sweep hits a north-west or a south-east corner of an obstacle the corresponding arcs are created and S is adjusted. Brucker also showed (cf. Brucker (1988) Theorem 2) that by ordering S according to the sum of the x-coordinate and the y-coordinate of their north-west corner one gets the same ordering as with the lexicographical ordering. However, his proof will not be valid anymore if we unify rectangles or stretch them. To overcome this problem we can apply the unifying and stretching *after* applying the network construction algorithm. Now unifying respective stretching means removing vertices from the network and replacing every arc entering or leaving these vertices with their counterparts entering or leaving the north west respective south east corner of the unified respective stretched obstacle.



(a) The collision



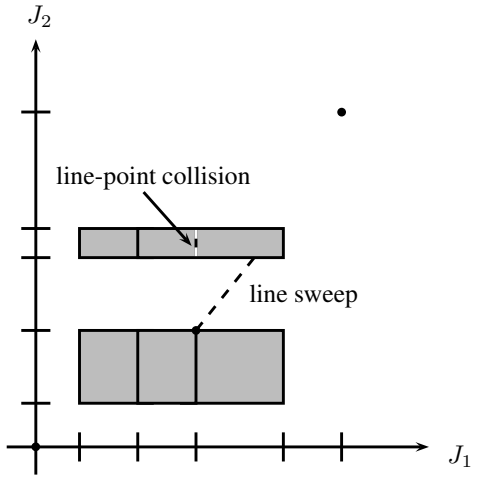
(b) Unified obstacle

Fig. 4: Handling of line-point collision

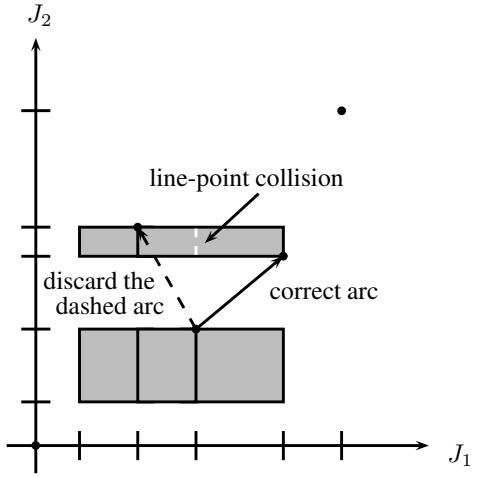
3.3 Handling the latency time

Now let $\tau^l > 0$: Grötschel et al. (2006) observed for the case $C_{\text{lp}} = C_{\text{ll}} = \emptyset$ that *BSSP* can be transformed into an equivalent problem with no latency if τ^l is sufficiently small, i.e. smaller than all processing times of restricted operations. Their modification increases the processing time of all general operations by τ^l and reduces the following processing time of a restricted operation accordingly.

This transformation is not valid anymore when collisions are present: Consider the example in Figure 7. Each job performs only one general operation. Both jobs share a general machine and there is a line-line collision $(D_{1,1}, D_{2,2})$. If we now optimize the problem over the transformed problem then we get the no-wait



(a) line sweep



(b) resulting arcs

Fig. 5: Postprocessing of line-sweep needed for line-point collisions

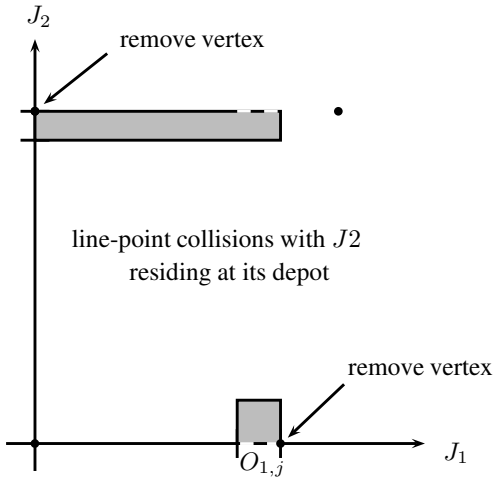


Fig. 6: Handling line-point collision involving a depot

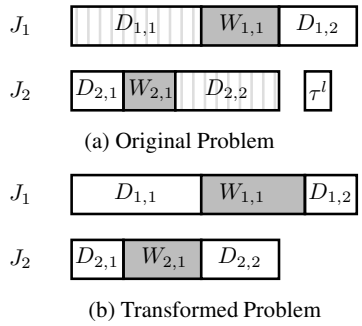


Fig. 7: τ^l cannot be removed

schedule. But this schedule is not feasible for the orginal problem due to the line-line collision $(D_{1,1}, D_{2,2})$.

In the following we will show how the latency time can be incorporated: let $W_{1,j}, W_{2,j'}$ be two general operations. We have to take care that at least τ^l time units have passed after $W_{1,j}$ is finished before the processing of $W_{2,j'}$ can start, and vice versa. We will concentrate on the first situation since the other one works analogously.

If the next restricted operation $D_{1,j+1}$ does not induce a line-line collision with $W_{2,j'}$ we can "locally" increase the general time of $W_{1,j}$ stretching the rectangle defined by $W_{1,j}$ and $W_{2,j'}$ to the right. If $D_{1,j+1} \geq \tau^l$ we can increase the rectangle's width by τ^l. Note that this increase can result in creating arcs going back to the past, which means that a target vertex has a smaller x or smaller y coordinate than the starting vertex. Similar to the handling of line-point collision we will discard all these arcs. Otherwise we can only increase the width by $D_{1,j+1}$. However, 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 line sweep 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 8).

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 line sweep 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 increased by τ^l and the weight of the second one by $\max\{\tau^l - p^d_{1,j+1}, 0\}$. Note that, if some of the rectangles have been unified, e.g. in the line-point

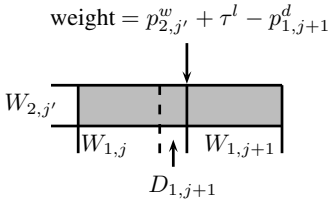


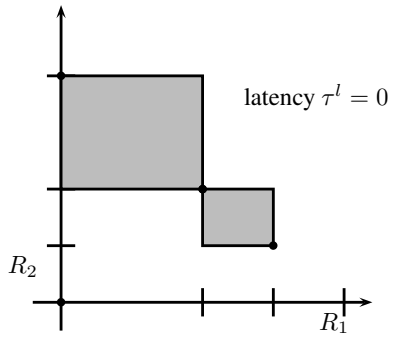
Fig. 8: Integrating latency

collision handling step, then the weight changes of the non-existent arcs will not be carried out.
Figure 9 illustrates the two job graph for instances corresponding to Figure 7. The first picture shows an instance without latency - whereas the other two include latency of 2 time units. In the central picture we used the invalid transformation where the processing times of general operations are increased and the processing times of restricted operations are decreased by τ^l. The third pictures shows our approach. The important difference is that the size of the rectangle $(D_{1,1}, D_{2,2})$ is reduced in the invalid transformation. This leads to the situation that the invalid schedule where each jobs processes its path without any waiting times becomes feasible. Due to the line-line collision $(D_{1,1}, D_{2,2})$ this schedule is indeed infeasible for the problem with latency time. Since our transformation never reduces the size of any rectangle this situation cannot occur.

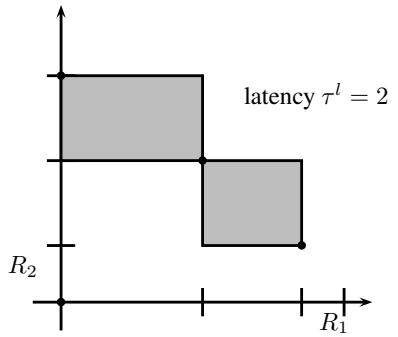
We finally state the main result of this section:

Proposition 1 *BSSP with $|R| = 2$ can be solved in $\mathcal{O}(n \log n)$ where n is the number of operation pairs sharing a common resource, i.e. a general machine or a line-line collision.*

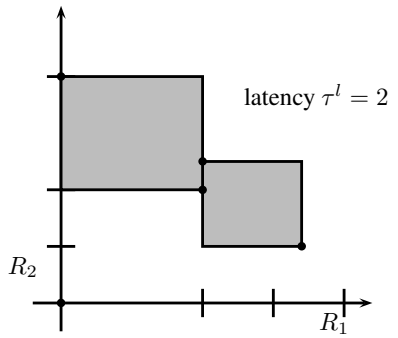
Proof Define an obstacle for every operation pair sharing a common resource with all modifications subject to collisions and latency as described above. This requires $\mathcal{O}(n)$ operations. Apply Brucker's algorithm to construct the network. This works in $\mathcal{O}(n \log n)$ (see Brucker (1988)) 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 $\mathcal{O}(n)$ steps. Finally, solve a shortest-path problem in the network. When applying topological sort and dynamic programming this requires $\mathcal{O}(n)$ steps. □



(a) Original Problem



(b) Invalid Transformation



(c) Correct Transformation

Fig. 9: latency handling comparison

4 NP-hardness of three jobs

We will reduce the even-odd partition problem to an instance of *BSSP* that is described by three jobs and one general machine. Furthermore the instance we use for the reduction is without any collisions or latency. Therefore this instance is 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 job each. Therefore all the operations of a job scheduled on his own restricted machine are conflict-free with respect to the other two jobs. Only the forth machine (corresponding to the general machine) has operations of all of the three jobs assigned. Hence this subproblem of job-shop scheduling has a very simple structure and the hardness result of Sotskov and Shakhlevich

(1995) for $(J3 \mid n = 3 \mid C_{\max})$ cannot be applied to the instances we use. Their result heavily relies on the fact that all three machines have to process operations of all three jobs and so the alternating chain structure that we use cannot be incorporated by their proof. When considering optimal solutions we can by Lemma 1 (or equivalently by the fact that the instance we use is a job-shop instance) concentrate on active schedules.

Definition 1 Even-odd partition

GIVEN: A multi-set $B := \{e_1, e_2, ..., e_{2n-1}, e_{2n}\}$ of $2n$ positive 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 NP-hardness of this problem already appeared in Garey and Johnson (1979). In order to simplify some bounds that are derived in proofs of this section we will use a slightly modified variant of even-odd partition: the pairs $\{e_{2i-1}, e_{2i}\}$ additionally have to fulfill $e_{2i-1} - e_{2i} \leq e_{2i}$. Note that this is indeed no restriction since any instance of even-odd partition can be transformed into an equivalent one that fulfills the additional requirement by defining the $2n$ positive integers as $\bar{e}_i = e_i + \sum_{j=1}^{2n} e_j$.

Definition of the *BSSP* instance

Let I be an instance of the even-odd partition problem. The instance J of *BSSP* consists of three jobs J_1, J_2 and J_3. For each pair (e_{2i-1}, e_{2i}) of integers in I $(i \in \{1, \ldots, n\})$ we define by Figure 10 a corresponding sequence of operations T_j^i for each of the three jobs j (T_j^i is called block with index i of job j in the following).

In this setting W_{jk}^i denotes general operations and D_{jk}^i restricted operations, $\delta_i := e_{2i-1} - e_{2i}$ and $H > 8E$, where $E = \sum_{i \in B} e_i$. The full sequence of operations for J_j is then defined by the sequence of blocks $T_j^1, T_j^2 \ldots T_j^n$. Note, that for the ease of readability we just skipped all general and restricted operations of length 0. These would actually occur in our instance whenever an operation of some type follows an operation of the same type.

The basic idea of our construction relies on the proof of Sotskov and Shakhlevich (1995) for $(J3 \mid n = 3 \mid C_{\max})$: All blocks of J_1 and J_2 are synchronized, i.e. in solutions that are relevant for our proof all blocks T_1^i and T_2^i end at the same point of time. The jobs J_1 and J_2 together represent the set B' of the even-odd partition. Moreover, the duration of all operations of T_1^i and T_2^i encodes whether e_{2i} or e_{2i-1} is added to B': When e_{2i} is added to B' then the duration of T_1^i and T_2^i is $12 \cdot H + e_{2i}$ otherwise it is $12 \cdot H + e_{2i-1}$. Job J_3 represents $B \setminus B'$. Here the duration of T_3^i is $12 \cdot H + e_{2i-1}$ if e_{2i} is added to B' and thus e_{2i-1} is added to $B \setminus B'$ otherwise it is $12 \cdot H + e_{2i}$. Figure 11 characterizes the situation that e_1 is added to B' and e_2 to $B \setminus B'$. Figure 12 describes the other possibility. We will show in our proof that for active schedules of weight $12 \cdot n \cdot H + E/2$ just these two situations are relevant. Observe that for blocks with index $i > 1$, the sequences T_1^i (resp. T_2^i) and T_3^i will be a little bit displaced with respect to each other. However, we will show that, due to the above setting of H, this displacement does not lead to infeasible schedules and so the even-odd partition instance and the *BSSP* instance correspond to each other. Note that even though the basic idea relies on Sotskov and Shakhlevich (1995), the construction we present differs in significant points from their approach. Moreover, the proof we present is simpler.

Scheduling the general operations 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 general machine is called block structure B_{1i} (cf. Figure 11). 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 12). The last two lines in each of the two figure improve the readability. If all blocks T_1^i, T_2^i and T_3^i are scheduled as B_{1i} or B_{2i} and if, for each $i = 1, \ldots, n$ and each machine, the last general operation of a bloc precedes the first general operation of the next one, we call such a schedule *useful*.

As we will show for getting the connection between even-odd partition and *BSSP* it suffices to consider active schedules with $C_{\max} = 12 \cdot n \cdot H + E/2$. For an instance J we can immediately get a lower bound of $12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i}$ by looking at the minimal total time for all operations of J_3 to be finished. If all operations are scheduled according to B_{2i} for all i we get that J_1 and J_2 finish their last oparation at $12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i}$.

Lemma 2 *Scheduling all general operations of J_2 and J_3 in the trivial order, i.e.*

$$(W_{21}^i, W_{22}^i, W_{23}^i, W_{31}^i, W_{32}^i, W_{24}^i),$$

can be done feasibly without causing any operation to wait (not taking J_1 into account).

Job	operations with their processing times of sequence T_j^i						
J_1	W_{11}^i e_{2i-1}	D_{11}^i H	W_{12}^i e_{2i}	D_{12}^i $3.75H + \delta_i$	W_{13}^i δ_i	D_{13}^i $2.5H$	W_{14}^i $2H - e_{2i-1}$
J_2	W_{21}^i H	D_{21}^i e_{2i-1}	W_{22}^i $2H$	W_{23}^i $2H$	D_{22}^i $4H$	W_{24}^i H	D_{23}^i $2H - e_{2i-1}$
J_3	D_{31} $6H + 2e_{2i-1}$	W_{31}^i H	W_{32}^i H	D_{32}^i $4H - e_{2i-1} - \delta_i$			

Fig. 10: The corresponding sequence of operations

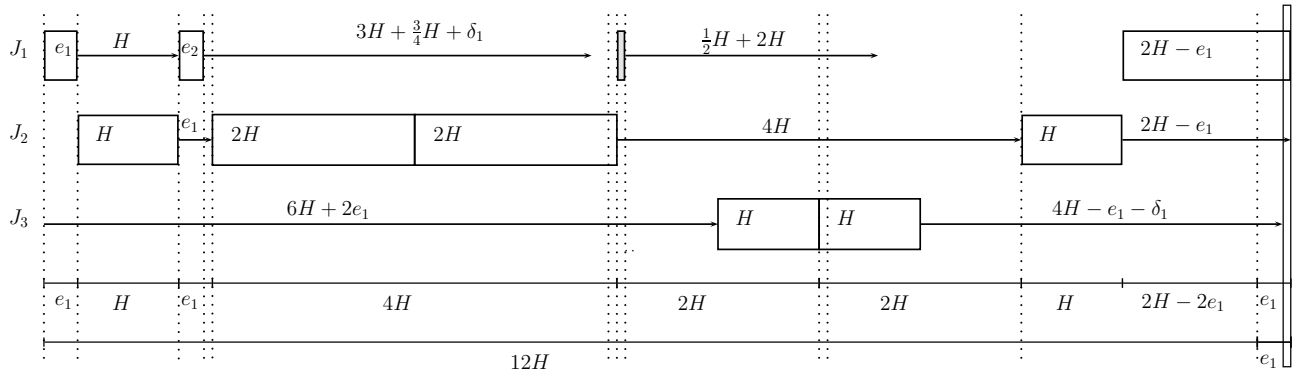


Fig. 11: Important types of schedules - Block Structure B_{11}

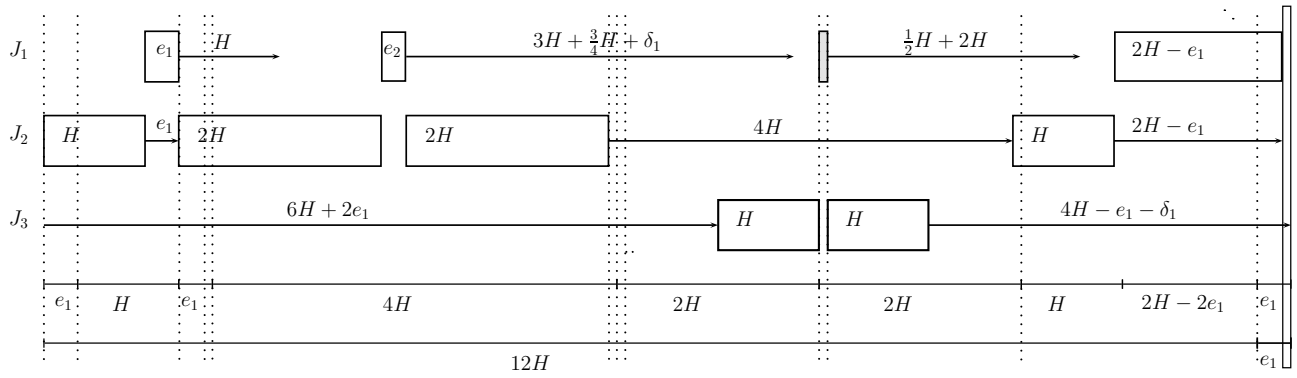


Fig. 12: Important types of schedules - Block Structure B_{21}

Proof We will consider the two jobs separately: If we schedule all operations of J_2 consecutively we get that W_{23}^i ends at $5H + e_{2i-1} + 12H(i-1)$. Operation W_{24}^i starts exactly $4H$ time units later.
If we schedule all operations of J_3 consecutively we get that W_{31}^i starts at $6H + 2e_{2i-1} + 12H(i-1) + \sum_{j=1}^{i-1} e_{2j}$ and operation W_{32}^i ends exactly $2H$ time units later.
Therefore, the difference between the starting of W_{31}^i and the finishing of W_{23}^i, i.e. $H + e_{2i-1} + \sum_{j=1}^{i-1} e_{2j}$, can be bounded as follows:

$$H + e_{2i-1} \leq H + e_{2i-1} + \sum_{j=1}^{i-1} e_{2j} \leq H + E \tag{1}$$

The difference between the starting time of W_{24}^i and the finishing time of W_{32}^i, i.e. $H - e_{2i-1} - \sum_{j=1}^{i-1} e_{2j}$, can be bounded from above (resp. from below) using $H > 8E$ by:

$$\frac{3}{4}H \leq H - e_{2i-1} - \sum_{j=1}^{n} e_{2j} \leq H - e_{2i-1} \tag{2}$$

So all operations of J_2 and J_3 can be scheduled in the trivial order without any conflicts on the general machine, furthermore all operations are processed without any waiting time in between. □

The following observation shows that in any schedule of $C_{\max} \leq 12 \cdot n \cdot H + E$ the operations of J_2 and J_3 have to be scheduled according to the trivial order.

Observation 1 *Suppose that they are scheduled according to the trivial order: by postponing the starting time of any of the operations of J_2 by $E+1$ time units a resulting schedule would have $C_{max} > 12 \cdot n \cdot H + E$. The same is true if one postpones any of the operations of J_3 by $E_0 := \sum_{i=1}^{n} e_{2i-1} + 1$ time units. This implies that the order of general operations of J_2 with respect to J_3 and vice versa has to be fixed to the trivial order for getting a schedule with $C_{max} \leq 12 \cdot n \cdot H + E$: otherwise the operations W_{31}^i and W_{32}^i cannot both be scheduled between W_{23}^i and W_{24}^i. If W_{32}^i is scheduled after W_{24}^i we get that the starting time of W_{32}^i is postponed by more than H time units contradicting E_o. If W_{31}^i is scheduled before W_{23}^i then W_{23}^i has to be postponed by at least $H > E + 1$ time units, again a contradiction.*

Lemma 3 *If, for all i, all operations are scheduled according the block structure B_{1i} and B_{2j}, 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 First observe that every schedule which consists only of block structures B_{1i} and B_{2j} is active. Furthermore, $C_{\max} \geq 12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i}$ is a lower bound due to J_3. Let us first consider the schedule defined by using only block B_{2i}. The finishing time of job J_1 and J_2 is $n \cdot 12H + \sum_{i=1}^{n} e_{2i}$ and the finishing time of J_3 is $n \cdot 12H + \sum_{i=1}^{n} e_{2i-1}$. This already gives us the required bound. We now show that such a schedule is feasible. The ending time of W_{32}^i equals:

$$8H + 2e_{2i-1} + \delta_i + (i-1) \cdot 12H + \sum_{j=1}^{i-1} e_{2j-1}$$

The starting time of W_{24}^i equals:

$$9H + 2e_{2i-1} - \delta_i + (i-1) \cdot 12H + \sum_{j=1}^{i-1} e_{2j}$$

Therefore the difference between these two equals:

$$H - 2\delta_i - \sum_{j=1}^{i-1} \delta_j$$

Obvious bounds that are used later for this term are:

$$0 \leq H - E \leq H - 2\delta_i - \sum_{j=1}^{i-1} \delta_j \leq H \tag{3}$$

Now consider the difference between the starting of W_{31}^i and the ending time of W_{23}^i:

$$\begin{aligned} &(i-1) \cdot 12H + \sum_{j=1}^{i-1} e_{2j-1} + 6H + 2e_{2i-1} \\ &- \left((i-1) \cdot 12H + \sum_{j=1}^{i-1} e_{2j} + 5H + e_{2i-1} + e_{2i}\right) \\ &= H + \sum_{j=1}^{i-1} \delta_j + \delta_i \end{aligned}$$

Therefore we obtain

$$0 \leq H \leq H + \sum_{j=1}^{i-1} \delta_j + \delta_i \leq H + E \tag{4}$$

Also note that the restricted operation D_{13}^i finishes before W_{24}^i and thus W_{14}^i can start immediately after W_{24}^i. We conclude that scheduling all operations according to block structure B_{2i} leads to a feasible active schedule. Let us now consider the case where all operations are scheduled according to block structure B_{1i}. Here the finishing time of J_1 and J_2 is $n \cdot 12H + \sum_{i=1}^{n} e_{2i-1}$ and the finishing time of J_3 is $n \cdot 12H + \sum_{i=1}^{n} e_{2i}$. For the difference between the starting of W_{24}^i and the ending of W_{32}^i we obtain

$$\begin{aligned} &(i-1) \cdot 12H + \sum_{j=1}^{i-1} e_{2j-1} + 9H + 2e_{2i-1} \\ &- \left((i-1) \cdot 12H + \sum_{j=1}^{i-1} e_{2j} + 8H + 2e_{2i-1}\right) \\ &= H + \sum_{j=1}^{i-1} \delta_j \end{aligned}$$

This leads to

$$0 \leq H \leq H + \sum_{j=1}^{i-1} \delta_j \leq H + E \tag{5}$$

For the difference between the starting of W_{31}^i and the ending of W_{23}^i we get

$$\begin{aligned} &(i-1) \cdot 12H + \sum_{j=1}^{i-1} e_{2j} + 6H + 2e_{2i-1} \\ &- \left((i-1) \cdot 12H + \sum_{j=1}^{i-1} e_{2j-1} + 5H + 2e_{2i-1}\right) \\ &= H - \sum_{j=1}^{i-1} \delta_j \end{aligned}$$

leading to

$$0 \leq H - E \leq H - \sum_{j=1}^{i-1} \delta_j \leq H \tag{6}$$

Again, D^i_{13} ends before W^i_{24} does so. Therefore using only block structure B_{1i} gives a feasible active schedule. Analyzing these two extremal situations is sufficient. If one considers a schedule that consists of both types of blocks (i.e. B_{1i} and B_{2i}) we get by similar calculations as above that the resulting schedule is feasible and that the Bounds 3, 4, 5 and 6 are still valid. Let J be the set of indices such that B_{1i} is chosen and $\bar{J}$ its complement. Then we get that the last operation of J_3 will end at $12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i} + \sum_{i \in \bar{J}} \delta_i$ and that the last operations of J_1 and J_2 will end at $12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i} + \sum_{i \in J} \delta_i$.

In any of the above cases the gaps between W^i_{31} and W^i_{23} and between W^i_{32} and W^i_{24} are in the interval

$$[H - E, H + E] \tag{7}$$

□

Lemma 4 *In any active schedule of a length smaller than or equal to $12 \cdot n \cdot H + E$ the operation W^i_{14} has to be performed directly after operation W^i_{24}.*

Proof It is not possible to schedule W^i_{14} between W^j_{21} and W^j_{22} or between W^j_{22} and W^j_{23} (for any j) since this would cause operation W^j_{23} to wait longer than $E + 1$ time units. Scheduling it between W^j_{31} and W^j_{32} is impossible since this would cause W^j_{32} to wait for more than H time units. Due to Bound 1 derived in Lemma 2 scheduling W^i_{14} between W^j_{23} and W^j_{31} would delay the starting time of W^j_{31} by more than E_0 time units. Scheduling it between W^j_{32} and W^j_{24} is impossible since it would delay the starting time of W^j_{24} by more than $E + 1$ time units (cf. Bound 2 of Lemma 2). Scheduling W^i_{13} after W^j_{24} followed by W^i_{14} would also result in a violation of the bound on C_{max}, since W^{i+1}_{21} would be postponed by more than $E + 1$ units of time. Hence, W^i_{14} is scheduled after W^i_{24} without any other general operation in between.

It remains to show that this is done without any time-lag. Assume that there is a time-lag. Since the schedule is active D^i_{13} finishes after W^i_{24} does so. This can only be caused by scheduling W^i_{13} after W^i_{32}. If it is scheduled before W^i_{32} we get that D^i_{13} ends before W^i_{24} does so: the sum of the processing time of W^i_{32} and W^i_{24} equals $2H$. By Bound 2 of Lemma 2 we know that the difference between the ending time of W^i_{32} and the starting time of W^i_{24} is greater than or equal to $\frac{3}{4}H$. Note that this bound is valid if all operations of J_2 and J_3 are scheduled in the trivial order. Observation 1 however implies that none of the operations of J_3 can be postponed by more than E_0 time units. So the the difference between the ending time of W^i_{32} and the starting time of W^i_{24} is indeed greater than or equal to $\frac{3}{4}H - E_0$ (postponing only operations on J_3 minimizes the gap). Still there is enough time for W^i_{13} and D^i_{13} to finish before W^i_{24} does so.

Assume that W^i_{13} is scheduled after W^i_{32}. This implies that the earliest possible starting time of W^{i+1}_{21} is $\frac{1}{2}H + 4H + \delta_i - e_{2i-1}$ time units after the starting time of W^i_{13} (by bounding the processing time of W^i_{13}, D^i_{13} and W^i_{14}). But this already gives a delay of at least $E + 1$ time units on J_2 contradicting the bound on C_{max}. □

The above lemma implies that the jobs J_1 and J_2 are synchronized in active schedules with $C_{\max} \leq 12 \cdot n \cdot H + E$. This means that the operations W^i_{14} and D^i_{23} end at the same time.

Lemma 5 *In any active schedule of length smaller than or equal to $12 \cdot n \cdot H + E$ the operation W^i_{13} is scheduled exclusively between W^i_{31} and W^i_{32} or directly before W^i_{31} on the general machine.*

Proof Assume that any other general operation $\tilde{w}$ besides from W^i_{13} is scheduled between W^i_{31} and W^i_{32}. If $\tilde{w}$ has a length greater or equal to H, W^i_{32} would be postponed for more than E_o time units, a contradiction to Observation 1. Operation W^i_{12} remains a possible candidate for being scheduled in this position. Since the earliest possible starting time for general operation W^i_{14} is now $6.25H + 2\delta_i$ time units after the ending time of W^i_{12}, together with Lemma 4 we get a contradiction: operation D_{23} starts at the same time as W^i_{14} does so and this corresponds to a delay of more than $E + 1$ time units. Operation W^i_{11} can be ruled out by similar arguments.

Scheduling W^i_{13} after W^i_{32} was already ruled out in the proof of Lemma 4. Assume that W^i_{13} is scheduled before W^i_{31} but not directly before it, i.e. some other operation is scheduled in between. Since the order of general operations of J_2 and J_3 is fixed with respect to each other W^i_{23} is always scheduled before W^i_{32}. Thus, we can assume w.l.o.g. that W^i_{13} is scheduled directly before W^i_{23}. By Lemma 4 job J_1 and J_2 finished block $i - 1$ simultaneously. Thus, scheduling W^i_{13} after W^i_{22} would postpone W^i_{23} by at least $W^i_{11} + D^i_{11} + W^i_{12} + D^i_{12} + W^i_{13} - W^i_{21} - D^i_{21} - W^i_{22} > E + 1$ time units which would imply $C_{\max} > 12 \cdot n \cdot H + E$. Note that scheduling any other operation between W^i_{13} and W^i_{22} leads to even larger violations. □

Lemma 6 *If an active schedule S of length smaller than or equal to $12 \cdot n \cdot H + E$ exists, we can find an useful schedule with total length not larger than the length of S.*

Proof By Lemma 5 we have to distinguish the following two cases:

Case 1. W^i_{13} is scheduled directly before W^i_{31}: if W^i_{12} is scheduled after W^i_{22}, W^i_{31} is postponed by more than $E_o + 1$ time units due to the total processing time of W^i_{12}, D^i_{12} and W^i_{13}. If W^i_{11} is scheduled after W^i_{21} and directly followed by W^i_{12}, operation W^i_{22} is postponed by more than $H > E+1$ units of time, since the general machine has to be idle throughout the processing time of D^i_{11}. 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_{22}) **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} the total processing time of the operations of T^i_1 and T^i_2 decreases, in the first case by δ_i in the second case by e_{2i-1}. Furthermore this sequence remains feasible by Lemma 3 and yields a useful schedule. □

Lemma 7 *C_{max} is greater than or equal to $12 \cdot n \cdot H + E/2$.*

Proof By considering only J_3 we get a lower bound of $12 \cdot n \cdot H + \sum_{i=1}^{n} e_{2i}$. Assume that there is a schedule with makespan $< 12 \cdot n \cdot H + E/2$. By Lemma 6 there exists also an useful schedule S with this property. However if the operations of T^i_1 and T^i_2 in the useful schedule need a processing time of $12H + e_{2i}$ then the operations of T^i_3 need a processing time of $12H + e_{2i} + \delta_i$ and vice versa. Therefore either all operations of J_1 and J_2 together need a total processing time not less than

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

or all operations of J_3 together need a total processing time not less than this bound. This contradicts to $C_{\max} < 12 \cdot n \cdot H + E/2$. □

Theorem 1 *BSSP is already NP-hard for three jobs, one general machine and without any collisions or latency.*

Proof $\Longrightarrow$ Assume that the even-odd partition I has a solution with $\sum_{e_j \in B'} e_j = \sum_{e_j \in B \setminus B'} e_j$. Construct J in the following way: If $e_{2i} \in B'$ use B_{2i} otherwise use B_{1i}. Then $C_{\max} = 12 \cdot n \cdot H + E/2$
$\Longleftarrow$ Let a solution of J be given with $C_{\max} = 12 \cdot n \cdot H + E/2$. By Lemma 6 and Lemma 7 we can find an 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 jobs J_1 and J_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 7 we get that the completion time of job J_3 will be $12 \cdot n \cdot H + \sum_{e_j \in B \setminus B'} e_j > 12 \cdot n \cdot H + E/2$. □

We get the following corollary as a direct consequence of the previous theorem.

Corollary 1 *The single machine scheduling problem that asks for a makespan minimal schedule of three chains of operations with delays between the operations of a chain, i.e. $(1 \mid 3 - chains(l_{i,j}) \mid C_{max})$, is $\mathcal{NP}$-hard.*

5 A pseudo-polynomial algorithm and an FPTAS

In this section, we will show that *BSSP* with a fixed number of k jobs and a fixed assignment of these jobs to $h \leq k$ general machines can be solved in pseudo-polynomial time.

5.1 The pseudo-polynomial algorithm

Our algorithm uses a transversal graph approach as introduced to scheduling problems by Middendorf and Timkovsky (1999). We will construct a directed acyclic graph $G = (V, A)$ with unit weight arcs where a shortest path coincides with an optimal solution to *BSSP*.
For $i = 1, \ldots, k$ let $O_{i,0}$ with $p_{i,0} = 0$ be an artificial start operation. Our vertex set V consists of a terminal vertex T together with tuples

$$(i_1, t_1, \ldots, i_k, t_k, j_1, d_1, \ldots, j_h, d_h).$$

These vertices are called *transversals*, with the following meaning:

- $i_r \in \{0, \ldots, 2n_r + 1\}$ is the index of the operation currently being processed by job J_r.
- $t_r \in \{0, \ldots, p_{r,i_r}\}$ denote how long J_r has already worked on O_{r,i_r}.
- $j_l \in \{0, 1, \ldots, k\}$ is the index of the job which used general machine M_l last. Here $j_l = 0$ means, that the general machine has not be used so far.
- $d_l \in \{0, \ldots, \tau^l\}$ is the time general machine M_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

$$I_c := \{r \mid 0 < t_r < p_{r,i_r}\}$$

the set of jobs currently working and by

$$I_n := \{r \mid i_r < 2n_r + 1, t_r = p_{r,i_r} \text{ and if } O_{r,i_r+1} \text{ is a general operation, then } (j_{l(J_r)} = r) \vee (j_{l(J_r)} = 0) \vee (d_{l(J_r)} = \tau^{l(J_r)})\}$$

the set of jobs which can start their next operation. These jobs are characterized by the facts that their last operation did not start, none of their operations is currently processed and if their next operation is a general operation, then their attached general machine has to fulfill one of the following properties:

- The general machine itself was used the last time by the considered job.
- The general machine was not yet used at all.
- If another job used the source last, then the latency time for switching the jobs has to be passed.

Here $l(J_r)$ denotes the index of the general machine job J_r is attached to.
Denote by $\mathcal{I}$ the set of all job sets I with $I_c \subseteq I \subseteq I_c \cup I_n$ which can work in parallel. This means, I induces no line-line collision, includes at most one general operation per general machine and induces no line-point collisions with any job $r \notin I$. 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

- $i'_r = i_r + 1$, for $r \in I \cap I_n$, and $i'_r = i_r$ otherwise
- $t'_r = t_r + 1$, for $r \in I \cap I_c$, $t'_r = 1$, for $r \in I \cap I_n$, and $t'_r = t_r$ otherwise
- $j'_l = r$ if $r \in I \cap I_n$ with J_r starts a general operation on M_l, and $j'_l = j_l$ otherwise
- $d'_l = 0$ if $r \in I \cap I_N$ with J_r starts a general operation on M_l, and $d'_l = d_l + 1$ if no job in I with $l(J_r) = l$ processes a general operation

So every operation of a job in I is processed for one time unit. Every vertex with $i_r = 2n_r + 1$ for all jobs will be connected with the terminal vertex T by an arc of zero weight. Note that the construction of the arcs guarantees that the schedules corresponding to paths of the transversal graph are non-preemptive.

Remark 2 In the proof of Lemma 1 of Section 2 we argued that the staring and completion times of all operations of active schedules are integer, therefore it is enough to consider integer time variables t_r and d_l in the transversals in order to represent active schedules..

Observation 2 *Every path in G from $(0, \ldots, 0)$ to T corresponds to a feasible schedule of BSSP where the makespan is the sum over the arc weights of P, i.e. the number of arcs. Moreover, every non-preemptive active schedule can be written in this form.*

We can extract the start times of the k-th operation of job r from a path P by summing up the arc weights of P between vertices with $i_r < k$.

Observation 3 *Let $p_{max} := \max_{i=1}^{k} \max_{j=1}^{2n_i+1} p_{i,j}$ be the maximal processing time, $\tau_{max} := \max_{l=1}^{h} \tau^l$ be the maximal latency, and $n_{max} = \max_{i=1}^{k} 2n_i + 2$ the maximal number of job operations (including the artificial start one). Then the number of vertices in G can be bounded by*

$$|V| \leq (n_{max})^k \cdot (p_{max} + 1)^k \cdot (\tau_{max} + 1)^h \cdot (k + 1)^h$$

and there are at most $2^k|V|$ arcs. Since $h \leq k$ and k is fixed, the size of G is pseudo-polynomial in the size of BSSP.

Theorem 2 *BSSP can be solved in pseudo-polynomial time when the number of jobs is fixed.* □

5.2 A fully polynomial time approximation scheme for BSSP

We close this section by providing a fully polynomial time approximation scheme (FPTAS) for *BSSP* with a constant number of jobs:

Theorem 3 *There is a fully polynomial time approximation scheme (FPTAS) for BSSP.*

Proof We will use a standard rounding argument which has been introduced by Ibarra and Kim (1975) for the knapsack problem and which is widely used in scheduling (see e.g. Agnetis et al. (2010)).
Let $\Delta := \max\{\max_{l=1}^{h} \tau^l, \max_{i=1}^{k} \max_{j=1}^{2n_i+1} p_{i,j}\}$ be the maximum processing time with respect to general and restricted operations and latency, and $N := \sum_{i=1}^{k} 2n_i + 1$ be the total number of operations. Define $Z := \epsilon \frac{\Delta}{N}$.
We construct a modified instance of *BSSP* with $\tilde{\tau}^l := \lceil \frac{\tau^l}{Z} \rceil$ and $\tilde{p}_{i,j} := \lceil \frac{p_{i,j}}{Z} \rceil$. By construction we have $\tilde{\tau}^l, \tilde{p}_{i,j} \leq 1 + N\frac{1}{\epsilon}$. Thus, all processing times of the modified problem are bounded by a polynomial in N and $\frac{1}{\epsilon}$. Therefore, the size of the transversal graph described above is also polynomially 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 operation 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 a semi-active schedule $\tilde{S^*}$ keeping the order of the operations on the machines. Here, a feasible non-preemptive schedule is semi-active if it is

not possible to decrease the starting time of any operation without changing the order of processing of any machine (cf. Pinedo (2008), Definition 2.3.5). Every active schedule is also semi-active.
Since rounding up enlarges the processing time of each operation by at most 1, we obtain $T(\tilde{S}^*) \leq \frac{T(S^*)}{Z} + N$. Thus, we have

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

□

We close this section with two remarks:

Remark 3 An FPTAS for $(J|n = k|C_{max})$ was given by Servakh and Shcherbinina (2006). Note that the FPTAS derived in Theorem 3 also applies to this case.

Remark 4 The results of this Section also hold for the following generalizations of *BSSP*:

- *BSSP* with release dates on the operations.
- *BSSP* with precedence constraints on the operations of different jobs.

This follows from the fact that the modifications needed for these generalizations can easily be incorporated in the transversal graph approach.

6 Conclusion

In this article we fully settled the approximability status of *BSSP* with a constant number of jobs 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 jobs 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 very special subproblem of job-shop scheduling.

Still interesting questions concerning *BSSP* with a non-constant number of jobs remain: what is the approximability (resp. inapproximability) status of this problem - the best known result is the strong $\mathcal{NP}$-hardness of Wikum et al. (1994). Are there special restricted types of collision (resp. constraints on the operations of the jobs) which still allow for polynomial time approximation schemes.

Acknowledgments

We thank the anonymous referees for their fruitful comments that helped to improve this paper considerably. The research was partly funded by the Austrian Science Fund (FWF): P23829.

References

A. Agnetis, M. Flamini, G. Nicosia, and A. Pacifici. A job-shop problem with one additional resource type. *Journal of Scheduling*, pages 1–13, 2010.

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

G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation; Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

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

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

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.

O.H. Ibarra and C.E. Kim. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *J. ACM*, 22(4):463–468, 1975.

M. Middendorf and V.G. Timkovsky. Transversal Graphs for Partially Ordered Sets: Sequencing, Merging and Scheduling Problems. *Journal of Combinatorial Optimization*, 3(4):417–435, 1999.

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

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

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.

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

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

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

V.V. Servakh and T.A. Shcherbinina. A fully polynomial time approximation scheme for two project scheduling problems. In *Information Control Problems in Manufacturing: Proc. 12th IFAC*, pages 129–134, 2006.

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

W. Welz. Route Planning for Robot Systems. Diplomarbeit, Technische Universität Berlin, 2010.

W. Welz and M. Skutella. Route Planning for Robot Systems. In *Operations Research Proceedings 2010*, pages 307 – 312. Springer, 2011.

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

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