The job shop scheduling problem with convex costs

Reinhard Bürgy [a]

Kerem Bülbül [b]

[a] *GERAD & Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal, Montréal (Québec), Canada*

[b] *Industrial Engineering, Faculty of Engineering and Natural Sciences, Sabanci University, Tuzla İstanbul, Turkey*

`reinhard.burgy@gerad.ca`
`bulbul@sabanciuniv.edu`

Abstract: The job shop scheduling literature has been dominated by a focus on regular objective functions – in particular the makespan – in its half a century long history. The last twenty years have encountered a spike of interest in other objectives, such as the total weighted tardiness, but research on non-regular objective functions has always been isolated and scattered. Motivated by this observation, we present a tabu search heuristic for a large class of job shop scheduling problems, where the objective is non-regular in general and minimizes a sum of separable convex cost functions attached to the operation start times and the differences between the start times of arbitrary pairs of operations. This problem definition generalizes a number of problems considered earlier in the literature. A particular notion of “critical paths” derived from the so-called timing problem is at the core of the proposed neighborhood definition exploited successfully in a tabu search algorithm. The computational results attest to the promise of our work.

Keywords: scheduling, job shop, non-regular objective, convex costs, tabu search

Acknowledgments: Part of the work of the first author was done during a stay at GERAD as a postdoctoral fellow at Polytechnique Montréal. R. Bürgy gratefully acknowledges the support of the Swiss National Science Foundation Grant P2FRP2_161720.

1 Introduction

The prevalence of custom manufacturing environments in a process layout is the key driver behind a vast literature on job shop scheduling problems. In this setting, a set of orders or jobs – each with a specific and potentially different recipe or route – is required to be processed by a set of resources with unary capacities. Each job is composed of a set of operations that need to be executed in a fixed given order, and each operation is to be performed on a specific resource for a prespecified duration of time. The main operational challenge facing a dispatcher is to determine a processing sequence for each of the resources given the routes and the processing requirements of the jobs with the perspective of optimizing an objective function of interest. In this regard, starting with the inception of the job shop scheduling problem in early 1960s the emphasis has long been on minimizing the completion time of the final operation, which tends to maximize throughput by minimizing the idle time in the schedule. In the common three field notation of Graham et al. (1979), this problem is described as $Jm//C_{\max}$, where m and $C_{\max}$ stand for the number of resources and the completion time of the final operation – also referred to as the makespan, respectively.

The problem $Jm//C_{\max}$ proved to be an archetype of hard combinatorial optimization problems and also served as a testbed for many algorithmic advances in the field (Jain and Meeran, 1999; Zobolas et al., 2008). Interestingly, this high research activity around $Jm//C_{\max}$ did not extend to other important objectives – in particular to those involving due dates until relatively recently. Custom manufacturing in job shops is by definition a make-to-order environment, and observing the job due dates is a significant dimension of customer satisfaction and service level in this setting. However, research with due date related objectives in the job shop environment only commenced in earnest with the two seminal papers by Singer and Pinedo on the job shop total weighted tardiness problem $Jm//\sum_j w_jT_j$ (Singer and Pinedo, 1998; Pinedo and Singer, 1999), where w_j and T_j denote the tardiness penalty per unit time and the amount of tardiness for job j, respectively. Observe that most of the subsequent influential research on $Jm//\sum_j w_jT_j$ appeared just in the last decade (Kreipl, 2000; De Bontridder, 2005; Essafi et al., 2008; Bülbül, 2011; Mati et al., 2011; González et al., 2012; Bierwirth and Kuhpfahl, 2017). Still, the total weighted tardiness objective $\sum_j w_jT_j$ is only a function of the job completion times and falls short of capturing increasingly important considerations in lean, efficient supply chains (Bülbül and Kaminsky, 2013), such as the intermediate inventory holding costs which result from idle times between consecutive operations of the same job. Not focusing on the operation completion times and the value added through the processing stages – as is the case with $Jm//C_{\max}$ and $Jm//\sum_j w_jT_j$ – may lead to substantially increased inventory holding costs. In-depth discussions around this subject are provided in Bülbül et al. (2004) and Bülbül and Kaminsky (2013). Another compelling reason to incorporate completion times, due dates, and cost functions at the level of individual operations into the problem definition is the so-called rescheduling problem (Avci and Storer, 2004; Bülbül and Kaminsky, 2013). A scheduling problem must frequently be solved starting from the partially realized schedule of a previous period or following a disruption, e.g., a machine breakdown. In such cases, resources and materials are already committed to certain operations at specific times, and the new schedule would not be allowed to move around such operations freely. These concerns can be reflected in the current scheduling problem by imposing the previously scheduled operation completion times as operation due dates and penalizing deviations from these.

From an application point of view, incorporating general convex cost functions is highly valuable as it allows for the representation of a rich set of performance measures and process features in a common framework. For instance, some production settings call for penalizing larger due date deviations disproportionately, resulting in superlinear earliness and tardiness costs. Employing such non-linear cost measures is for instance relevant in order to emphasize the essentialness of meeting a given due date (Cheng and Jiang, 1998) (almost) exactly. Perishability of the final product is another consideration giving rise to non-linearity (Amorim et al., 2013). This feature is present in various industries including the processed food and beverages sectors (Farahani et al., 2012), chemical and pharmaceutical industries (Neumann et al., 2002; Vidal et al., 2010), and blood banks (Prastacos, 1984). Perishability can also sometimes apply to intermediate goods. In the yogurt industry, for example, raw milk must be processed into its final products within strict time limits as the main ingredients are highly perishable at all production stages (Amorim et al., 2013). The management of perishability presents an important challenge in production scheduling, and it is usually modeled with hard maximum storage time constraints. In contrast, our scheduling model enables a more fine-grained control by charging storage costs that are convex in the storage time. This "soft" modeling approach is also related to another common modeling construct captured by our framework – soft precedence constraints. Although precedence constraints are usually specified as hard restrictions in scheduling models, some are actually "soft" in practice, i.e., they may be violated at some cost, particularly if such violations give rise to a better overall schedule. Applications of soft precedence constraints can be found in various settings including the scheduling of construction work (Jaskowski and Sobotka, 2012; Peng and Ouyang, 2012), sequencing in multi-agent (computer) systems (Cheng et al., 1997; Van Hoeve et al., 2007), and the planning of vessel fleet schedules (Fagerholt, 2004).

Motivated by the considerations above, we present a scheduling model that captures a large class of job shop scheduling problems. The objective is non-regular in general and minimizes a sum of separable convex cost functions attached to the operation start times and the differences between the start times of arbitrary pairs of operations. This problem definition – described as $Jm||\text{conv}$ in the scheduling notation and referred to as **JS–CONV** in the sequel – generalizes a number of problems considered earlier in the literature both with regular and non-regular objectives. These include the strongly $\mathcal{NP}-$hard job shop makespan, total weighted completion time, and total weighted tardiness problems, as well as the job shop just-in-time scheduling problem (with intermediate inventory holding costs), and render **JS–CONV** strongly $\mathcal{NP}-$hard. Following the presentation of our precise mathematical model in Section 2, we will give concrete examples of how our problem subsumes many frequently studied classical job shop scheduling problems in the literature.

The most common type of non-regular scheduling objective tackled in the job shop literature is minimizing the (weighted) sum of earliness and tardiness costs, at times also incorporating the intermediate inventory holding costs by charging penalties proportionally to the waiting times of the operations. Special cases and variants of this general *linear* objective have been studied in the literature. In particular, Beck and Refalo (2002, 2003); Chen and Luh (2003) take on the basic form of this objective function by restricting their attention to minimizing the total weighted earliness and tardiness. Baptiste et al. (2008) consider a more general version of this problem by assigning costs to the deviations from the due dates defined at the level of the individual operations. Ohta and Nakatanieng (2006) penalize the waiting times between the successive operations of a job and the earliness amounts, but do not allow for tardy jobs in their model. All three cost components appear in Sadeh (1996) and Brandimarte and Maiocco (1999). In the earlier work, each unit of time from the start time of an operation until the job completion time – or the due date if the job is early – accumulates cost in a linear fashion. However, in the later work, only the holding time of a job – as measured from the start time of its initial operation until its delivery time – incurs a cost at a constant rate. In both studies, earliness is captured implicitly within the holding cost term. In this stream of research with linear earliness, tardiness, and holding costs, Avci and Storer (2004) and Bülbül and Kaminsky (2013) present the most general problem definitions encompassing all of the problems discussed above. In both works, the fundamental requirement is that the timing problem – explained further down in this section – boils down to a linear program (LP) even though both sets of authors select a particular general version for their exposition of the material.

An early contribution to the job shop scheduling literature with a non-regular and non-linear objective is due to Kaskavelis and Caramanis (1998), who include weighted squared tardiness costs in addition to the linear holding costs between successive operations of jobs. The next three studies require a special emphasis because together with this paper they constitute the body of literature on generic job shop scheduling problem definitions, allowing for both non-regular and non-linear objectives and various processing features. In particular, Grimes and Hebrard (2015) recently considered a generic disjunctive machine scheduling problem within a constraint programming framework compatible with both regular and non-regular scheduling objectives, and problem characteristics such as maximal time lags between successive operations of the same job and sequence-dependent setup times. In Gélinas and Soumis (2005), each operation completion time is associated with a piecewise linear – and not necessarily monotone – cost function, and the overall objective is flexible enough to combine these functions in a min-max or min-sum form. The additional processing constraints that are taken into account are, however, limited to release times and deadlines. The general job shop scheduling model studied by Wennink (1995) in his PhD thesis is the closest existing model to ours. Wennink's model not only captures complicated processing features, such as generic time lags between two operations, sequence-dependent setup times, and multiple simultaneous machine assignments for an operation, but it also makes it possible to specify a generic convex piecewise-linear objective function. This objective function is very similar to that we address because we ask for integer start times, and our convex objective function can therefore be replaced by a convex piecewise-linear objective with integer breakpoints – see Section 3 for more details. The punch line of this discussion is that **JS–CONV** subsumes all job shop scheduling problem definitions pointed out up to this point – with the exception of Grimes and Hebrard (2015) and Gélinas and Soumis (2005) because these studies do not require the convexity of the objective, and Wennink (1995) who incorporates additional processing characteristics.

The theoretically and practically challenging nature of job shop scheduling problems has created a very rich literature on local search methods. We mainly restrict our attention to these in an effort to position our work with respect to the literature because from a methodological viewpoint our study falls into the realm of local search as well. In the domain of regular objectives, we only provide pointers to some of the more frequently cited influential papers, and then proceed to describe the research landscape in job shop scheduling with non-regular objectives in more detail.

Virtually all local search methods in the job shop environment benefit from decomposing the decisions into two phases. The first task at hand is to determine an operation processing sequence for all machines involved. In the second phase, the objective is to compute the optimal operation start/completion times for these fixed processing sequences. It turns out that the second phase – referred to as the *timing problem* – is in general a simple optimization problem solvable in polynomial time while the combinatorial explosion of the sequencing decisions is the main culprit in the difficulty of job shop scheduling. For regular objectives, the timing problem boils down to scheduling all operations as early as possible without violating the operation precedences within the jobs and the fixed operation processing sequences, and is modeled and solved as a longest path problem. The longest paths – also called *critical paths* – for the given fixed processing sequences are then leveraged to define the *neighborhood* of the current solution in a local search method. This idea has been successfully exploited for $Jm//C_{\max}$ (Van Laarhoven et al., 1992; Nowicki and Smutnicki, 1996; Balas and Vazacopoulos, 1998; Nowicki and Smutnicki, 2005; Peng et al., 2015) and $Jm//\sum_j w_j T_j$ (Kreipl, 2000; De Bontridder, 2005; Essafi et al., 2008; Bülbül, 2011; Kuhpfahl and Bierwirth, 2016) by several authors. The interested reader is referred to Vaessens et al. (1996) and Kuhpfahl and Bierwirth (2016) for an overview of the various neighborhood operators applied to $Jm//C_{\max}$ and $Jm//\sum_j w_j T_j$. The work of De Bontridder (2005) must receive a special mention here because similar to our work this author relies on a network flow model of the timing problem for an operation swap-based neighborhood definition embedded into a tabu search algorithm. The job shop model in focus incorporates generalized precedence relationships among operations and allows for positive end-to-start time lags, but is limited to minimizing the total weighted tardiness.

In the domain of regular scheduling objectives, the papers by Mati et al. (2011) and Bürgy (2017) deserve special credit for attacking job shop scheduling problems with a general regular objective. In particular, Mati et al. (2011) demonstrate that the most basic neighborhood of reversing a single arc on a critical path enhanced with a novel and fast move evaluation procedure gives rise to a local search method which performs on a par with custom-made algorithms for five common regular objectives. Bürgy (2017), on the other hand, addresses complex process attributes on top of a general regular objective. The author proposes a tabu search employing a neighborhood constructed by extracting a job and re-inserting it in a new position, which in some sense corresponds to generalizing the reversal of critical arcs to complex job shops. The method is tested with five different regular objectives in three standard scheduling environments: the conventional job shop, the job shop with sequence-dependent setup times, and the blocking job shop. The numerical results show that the proposed approach performs well in all tested cases. Both of these papers indicate that there is promise in pursuing algorithms applicable to a broad class of job shop scheduling problems without compromising from solution quality and time. We follow this lead in this paper.

As evident from the discussion up to this point, there is a plethora of research related to local search methods for job shop scheduling with regular objectives. However, there are only a few relevant contributions with non-regular specific or more general objectives. From a technical perspective, the availability of relatively efficient solution techniques for the timing problem is still the driving force behind this stream of work. Beck and Refalo (2002) design a tabu search algorithm, where a neighboring solution is obtained by reversing the execution order of two adjacent critical operations. The interesting component of this research is the way operation criticality is detected. The authors resort to constraint programming tools to this end. The numerical results are not really favorable in comparison to exact methods. The basic blueprint of the next set of three studies (Brandimarte and Maiocco, 1999; Avci and Storer, 2004; Wennink, 1995) is identical – the neighborhood definition is based on the timing problem formulated and solved as a maximum cost network flow problem, while the job shop environments and the associated objectives under consideration are quite different as detailed earlier in this section. Brandimarte and Maiocco (1999) apply a pure local search algorithm by inverting an arc carrying a positive flow in the network flow solution in each iteration. This is just a generalization of reversing a critical arc in the longest path calculations in the case of regular objectives. Avci and Storer (2004) take a similar path and analyze further properties of the maximum cost network solution in order to develop more compact adjacent pairwise interchange neighborhoods that are applicable to a broad class of scheduling problems with both regular and non-regular linear objectives. Recall that Wennink (1995) lays out a more general modeling framework with a generic convex piecewise-linear objective function. He demonstrates that the timing problem does still boil down to a maximum cost network flow problem and develops a rich set of theoretical properties based on the timing solution. An insertion-based neighborhood is examined in addition to the adjacent pairwise interchange neighborhood, and both are embedded into a tabu search. However, the computational study remains restricted to the makespan objective. The reason for this choice is rooted in the author's statement: "Much research, however, has still to be performed before such methods can be made time efficient... Furthermore, although a selection can be evaluated in polynomial time by solving a maximum cost flow problem, the time requirement is still so high that a straightforward implementation would spend too much time in evaluating each of the many selections that must be considered." In fact, the concern that solving an LP or a network flow formulation of the timing problem is far from warranting a fast exploration of a sufficient number of solutions in the feasible region resulting in a poor final solution quality is a recurring theme in these papers. A further complicating factor in the case of piecewise-linear convex objectives is that a straightforward modeling approach requires an additional arc for each linear segment in the objective function – see (Wennink, 1995, p.73) and (Ahuja et al., 2003, Section 4). This would blow up the network size and render the solution time of the timing problem totally unacceptable within an iterative algorithm. The development in Section 3 establishes that the timing problem with convex cost functions can still be solved efficiently without an extended network, and this is a major contribution of our paper as it paves the way for building local search algorithms for job shop scheduling problems with more general processing features and objectives.

An overview of the alternate approaches applied to job shop scheduling problems with non-regular objectives completes the review of the relevant literature. We are only aware of three exact methods in this domain. The recent work of Grimes and Hebrard (2015) is a pure constraint programming algorithm, which combines a number of generic search techniques. While the basic approach is generic, some problem-specific knowledge is incorporated to enhance the computational effectiveness for certain problem types. The numerical tests on some standard job shop problems, including the job shop with linear earliness/tardiness costs, demonstrate a performance competitive with custom state-of-the-art methods on small to medium-sized instances; however, the authors concede that scalability to larger instances may be an issue. Next is a hybrid exact algorithm by Beck and Refalo (2003). At any node of the search tree, a linear timing problem yields the optimal operation start times by ignoring any resource restrictions. If resource conflicts ensue, the search resorts to inference methods based on constraint programming for resolution and branching decisions. The results are fairly successful on instances with up to 200 operations if the due dates are not tight. In this case, relaxations high up the search tree are easily extended to full feasible solutions. The branch-and-price algorithm of Gélinas and Soumis (2005) completes the set of exact methods. The underlying pillar of this approach is a Dantzig-Wolfe re-formulation of the job shop scheduling problem under consideration, in which the precedence constraints are relaxed – i.e., kept in the master problem – and a pricing subproblem is solved for each machine. The advantage this method enjoys is that it attains optimal solutions for instances with up to 10 machines and 500 operations of a just-in-time scheduling problem similar to that studied in Baptiste et al. (2008) in one hour of computation time as long as the number of operations per job is kept small. It is also worth underlining that both Grimes and Hebrard (2015) and Gélinas and Soumis (2005) apply their solution techniques to linear problems only despite their more general modeling frameworks – see earlier in this section.

Non-regular objectives tackled through mathematical programming-based approaches are rare in the job shop scheduling literature. Other than the study by Gélinas and Soumis (2005), only Lagrangian relaxation-based heuristics (e.g., Kaskavelis and Caramanis, 1998; Chen and Luh, 2003; Baptiste et al., 2008) have appeared so far. An absolute solution quality assessment of these algorithms against the state-of-the-art is not really possible because the authors either compare alternate Lagrangian methods or employ dispatch rules for benchmarking purposes. One notable exception is the presentation of the gaps between the upper bounds obtained from a primal heuristic and the lower bounds in Baptiste et al. (2008). In this case, the gaps are quite large for most of the instances included, and overall, these studies offer no compelling evidence that their methods would be competitive with local search algorithms.

The final popular genre of algorithms in job shop scheduling is based on the shifting bottleneck paradigm – a special form of machine decomposition originally proposed for $Jm//C_{\max}$ (Adams et al., 1988). In job shop scheduling with non-regular objectives, the studies by Ohta and Nakatanieng (2006) and Bülbül and Kaminsky (2013) belong to this group. While the earlier shifting bottleneck algorithm is only focused on the specific problem at hand, Bülbül and Kaminsky (2013) leverage the dual information retrieved from the timing problem formulated and solved as an LP in a novel way to construct the machine subproblems. In computational testing, the authors demonstrate the value of the proposed approach with respect to alternate methods on several problems with intermediate holding costs. In Section 5.2, we demonstrate that our new local search for **JS–CONV** outperforms the shifting bottleneck heuristic of Bülbül and Kaminsky (2013).

We conclude this section by a summary of our contributions. By far the most important attribute of our work is its generality – both from the modeling and solution method perspectives, and this is not a point to be taken lightly. The proliferation of quite specific problem definitions and associated tailor-made, sophisticated, and highly parametric (meta-)heuristics based solution approaches, which do not easily translate or extend to other problems, is a significant shortcoming of the scheduling literature in general. See Bülbül and Kaminsky (2013) for a more in-depth discussion on the issue. Thus, our work joins a growing but still a small group of job shop scheduling studies with an emphasis on generality (Wennink, 1995; Gélinas and Soumis, 2005; Mati et al., 2011; Bülbül and Kaminsky, 2013; Grimes and Hebrard, 2015; Bürgy, 2017). Our numerical results on two different types of problems support our vision, and the extensive computational study on a just-in-time job shop scheduling problem with a non-linear objective in Section 5.3 is the first of its kind – Kaskavelis and Caramanis (1998) report results from just a few instances. Finally, we dispel Wennink's concerns and show that the timing problem with convex costs is amenable to quick solution times so that it may be embedded successfully into a meta-heuristic algorithm, paving the way for further research.

The paper is organized as follows. The next section formally describes **JS–CONV** and introduces a classical disjunctive graph formulation, which exposes the two-phase decomposable nature of job shop scheduling with the timing and sequencing components. The timing problem associated with **JS–CONV** is analyzed in Section 3, and an exact network flow-based solution approach is discussed. The solution of the timing problem is then leveraged in Section 4 to devise a tabu search for **JS–CONV** applied to two non-regular job shop scheduling problems – one with a linear objective and another with a non-linear objective. The value of the proposed approach is supported by the numerical results presented and discussed in Section 5. Some content is relegated to the appendix, which is provided as an online supplement.

2 The job shop scheduling problem with convex costs

In this section, we formally describe **JS–CONV** by introducing the necessary notation and then provide a disjunctive mathematical programming formulation and an associated disjunctive graph representation of the problem.

Consider a job shop with a set of m machines denoted by $\mathcal{M}$ and a set of operations $\mathcal{I}$ to be performed on these machines. All machines are available continuously from time zero onward, and a machine can execute no more than one operation at a time. Each operation $i \in \mathcal{I}$ needs a specific machine, say $M(i) \in \mathcal{M}$, for its non-preemptive execution of duration $p_i > 0$, where p_i is an integer. The set of operations is partitioned into a set of jobs $\mathcal{J}$ of cardinality n such that each operation $i \in \mathcal{I}$ belongs to exactly one job $J(i) = K$. Each job $K \in \mathcal{J}$ is represented as an ordered set of operations $\{K_1, K_2, \dots, K_{|K|}\}$, where K_r denotes the r-th operation of job K. The order of the operations in K specifies the sequence in which these operations must be processed, and two operations $i, j \in K$ are termed as *consecutive* if $i = K_r$ and $j = K_{r+1}$ for some $r,\ 1 \leq r < |K|$.

A vector $\boldsymbol{\alpha} = (\alpha_i \in \mathbb{Z}^* : i \in \mathcal{I})$ of non-negative integers, where α_i stands for the start time of operation $i \in \mathcal{I}$, specifies a *schedule*. The integrality of the processing times and the operation start times can be regarded as a precision constraint reflecting the smallest time unit in the scheduling process. A schedule $\boldsymbol{\alpha}$ is feasible if the operations of each job are processed in the given required sequence, and each machine executes at most one operation at any point in time. For any pair of distinct operations $(i, j) \in \mathcal{I} \times \mathcal{I}$, we allow to charge a non-negative cost of $f_{ij}(\delta_{ij})$ against the difference $\delta_{ij} = \alpha_j - \alpha_i$ of the start times of operations i and j, where $f_{ij} : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is a convex function. This applies to a subset $Q \subseteq \mathcal{I} \times \mathcal{I}$ of the pairs of operations. Similarly, an operation $i \in P \subseteq \mathcal{I}$ incurs a non-negative cost of $b_i(\alpha_i)$ if it starts processing at time α_i, where $b_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is convex. The total cost $c(\boldsymbol{\alpha})$ of the schedule $\boldsymbol{\alpha}$ is then calculated as $c(\boldsymbol{\alpha}) = \sum_{(i,j) \in Q} f_{ij}(\delta_{ij}) + \sum_{i \in P} b_i(\alpha_i)$.

Some remarks on the functions f and b are in order. First, we assume that they can efficiently be evaluated in $\mathcal{O}(1)$ time. Second, one could define problems where the total cost decreases as the start times of some operations approach infinity. We cannot handle this in our framework. In order to rule out this case, we assume that all functions f and b are bounded from below. Specifically, for all $(i, j) \in Q$, there exists a finite value $\delta_{ij}^{\min} \in \mathbb{R}$ such that $f_{ij}(\delta_{ij}) \geq f_{ij}(\delta_{ij}^{\min})$ for all $\delta_{ij} \in \mathbb{R}$, and for all $i \in P$, there exists a finite value $\alpha_i^{\min} \in \mathbb{R}_{\geq 0}$ such that $b_i(\alpha_i) \geq b_i(\alpha_i^{\min})$ for all $\alpha_i \in \mathbb{R}_{\geq 0}$. Third, without loss of generality, we assume that $\delta_{ij}^{\min}$ is non-negative. If $\delta_{ij}^{\min} < 0$, we can specify the same cost structure using the start time difference between j and i; that is, by introducing a function f_{ji} instead of f_{ij}.

The problem **JS–CONV** asks to find a feasible schedule $\boldsymbol{\alpha}$ with the minimum cost $c(\boldsymbol{\alpha})$ and can be formulated as the following disjunctive programming problem:

$$\text{minimize} \sum_{(i,j) \in Q} f_{ij}(\delta_{ij}) + \sum_{i \in P} b_i(\alpha_i) \tag{1a}$$

subject to:

$$\alpha_j - \alpha_i \geq p_i \text{ for all } i, j \in \mathcal{I} \text{ consecutive in some job,} \tag{1b}$$

$$\alpha_j - \alpha_i \geq p_i \text{ or } \alpha_i - \alpha_j \geq p_j \text{ for all } i, j \in \mathcal{I} \text{ with } M(i) = M(j),\ \ J(i) \neq J(j), \tag{1c}$$

$$\alpha_j - \alpha_i - \delta_{ij} = 0 \text{ for all } (i, j) \in Q, \tag{1d}$$

$$\alpha_i \in \mathbb{Z}^* \text{ for all } i \in \mathcal{I}. \tag{1e}$$

Any feasible solution $\boldsymbol{\alpha}$ of (1) specifies a set of non-negative integer start times α_i for all operations $i \in \mathcal{I}$ by (1e). Constraints (1b) ensure that the precedence relationships within a job are satisfied, and constraints (1c) state that any pair of operations sharing the same machine must not be executed in parallel. This is already valid for each pair of operations from the same job by (1b). Hence, we add a constraint of type (1c) only if $J(i) \neq J(j)$. Finally, together with constraints (1d), which relate the auxiliary variables $\boldsymbol{\delta}$ to the primary variables $\boldsymbol{\alpha}$, (1a) specifies the objective.

The generic definitions of the sets P and Q as well as the functions f and b render the formulation (1) flexible enough to accommodate quite different problem settings. In particular, **JS–CONV** subsumes many existing and frequently studied job shop scheduling problems. For instance, setting $Q = \emptyset$, $P = \{i \in \mathcal{I} \mid i = K_{|K|} \text{ for some } K \in \mathcal{J}\}$ and $b_i(\alpha_i) = \max\left(0, (\alpha_i - (d_K - p_i))c_K^{\text{tardy}}\right)$ for $i \in P$, where $K = J(i)$ and c_K^{tardy} is the unit tardiness cost for job K, captures $Jm//\sum_j w_j T_j$. The job shop total weighted completion time problem is obtained by simply specifying all due dates as zero. To model $Jm//C_{\max}$, we can add a common dummy terminal operation τ to all jobs $K \in \mathcal{J}$ with $p_\tau = 0$, and then define $Q = \emptyset$, $P = \{\tau\}$, and $b_\tau(\alpha_\tau) = \alpha_\tau$. Another existing example that fits into our framework is the job shop just-in-time scheduling problem with intermediate inventory holding costs (Bülbül and Kaminsky, 2013). This is one of the selected problems for

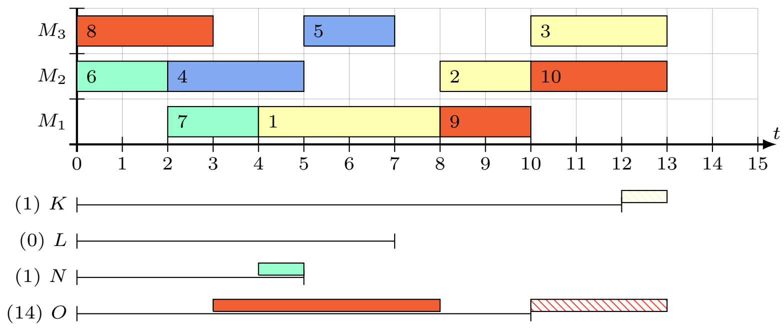


Figure 1: A feasible solution of the example with a total cost of 16. (top) Gantt chart of the solution. (bottom) The rectangles represent storage times / earliness values (filled) and tardiness amounts (hatched). A black line indicates the time period up to the due date of the associated job, and the total cost for each job is indicated on the left in brackets.

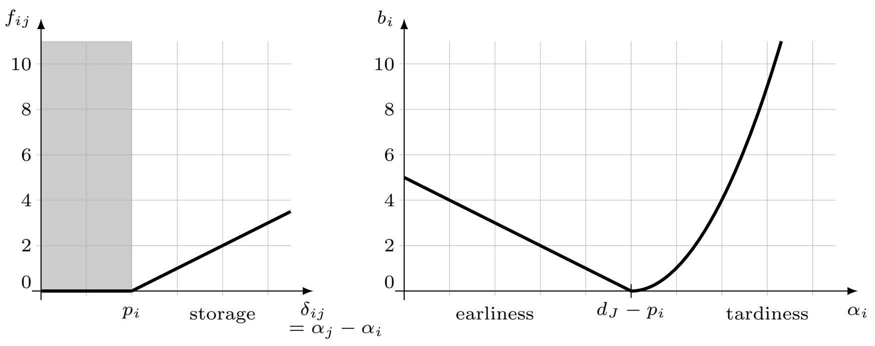


Figure 2: (left) The storage cost function $f_{ij}(\delta_{ij}) = \max(\delta_{ij} - p_i, 0)$ for some consecutive operations i and j in a job. By precedence constraints (1b), the values in the gray shaded area are not attained in any feasible solution. (right) Linear earliness and squared tardiness cost function for a job J, where $i = J_{|J|}$ is its last operation.

demonstrating the performance of our solution approach in our numerical study, and its mapping to **JS–CONV** is detailed at the start of Section 5.1. In addition to these common problem types, other interesting considerations can also be reflected in **JS–CONV**. To illustrate, recall that in most traditional scheduling models the only mechanism of prioritizing jobs is through objective function weights. In **JS–CONV**, we can additionally incorporate (i, j) into Q, where $i = K_{|K|}$ and $j = L_{|L|}$, and define an appropriate convex function $f_{ij}(\delta_{ij})$ to penalize the difference $\alpha_j - \alpha_i$. This is a more explicit way of specifying a "soft" priority for completing job L before job K. In another application, assume that an intermediate product is perishable between operations i and j, and waiting times between i and j are increasingly unacceptable. **JS–CONV** can cope with this concern by imposing an appropriate convex cost function $f_{ij}(\delta_{ij})$, where $(i, j) \in Q$. Generally speaking, **JS–CONV** can accommodate soft no-wait or maximum time lag constraints between pairs of operations in Q by a clever selection of the functions f.

We end this section by introducing a small illustrative example to be used in the remainder of the paper. The example consists of three machines $\mathcal{M} = \{M_1, M_2, M_3\}$, ten operations $\mathcal{I} = \{1, \ldots, 10\}$, and four jobs $K = \{1, 2, 3\}$, $L = \{4, 5\}$, $N = \{6, 7\}$, and $O = \{8, 9, 10\}$. The processing data can directly be read in the feasible solution depicted in the Gantt chart of Figure 1. For example, operation 2 is executed on machine M_2 and has a duration of 2 time units. Each job has a due date: $d_K = 12$, $d_L = 7$, $d_N = 5$, and $d_O = 10$. We consider a convex just-in-time objective and penalize storage, earliness, and tardiness. For all jobs, the storage and earliness costs are one per time unit and we add squared tardiness costs. Specifically, for all consecutive operations i and j in a job, define $f_{ij}(\delta_{ij}) = \max(\delta_{ij} - p_i, 0)$ (see Figure 2, left), and for the last operation $i = J_{|J|}$ of each job $J \in \mathcal{J}$, let $b_i(\alpha_i) = \max\left\{(d_J - p_i) - \alpha_i, (\alpha_i - (d_J - p_i))^2\right\}$ (see Figure 2, right). Note that if $\alpha_i = d_J - p_i$, then job J finishes just in time and no earliness and tardiness costs arise.

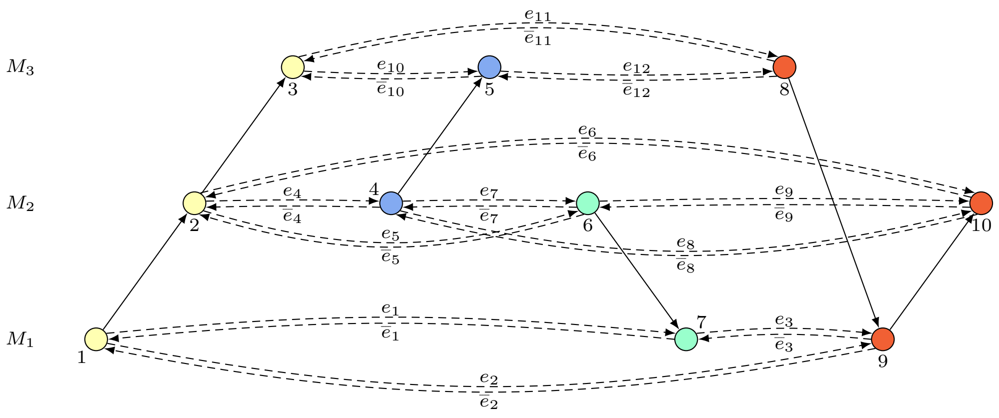


Figure 3: The disjunctive graph G of the illustrative example. The arc lengths are omitted.

2.1 Problem formulation on a disjunctive graph

The disjunctive formulation (1) can be mapped to a so-called disjunctive graph representation of the problem. Initially proposed by Roy and Sussmann (1964), this representation has been central to countless local search (and sometimes exact) approaches developed for job shop scheduling problems – we will follow suit in the next sections. In the particular disjunctive graph $G = (\mathcal{I}, A, E, \mathcal{E}, d)$ representation employed in this paper, $\mathcal{I}$ is the set of nodes, A is the set of *conjunctive arcs*, E is the set of *disjunctive arcs*, $\mathcal{E}$ is a family of disjunctive arc pairs representing the disjunctive structure of the problem, and d is an arc length vector. Specifically, graph G is built as described next.

Each operation $i \in \mathcal{I}$ is represented by a node, and we identify a node with the operation it represents. For all consecutive operations i and j in a job, a so-called conjunctive arc (i, j) of length $d_{ij} = p_i$ is incorporated into the set A. These arcs capture the precedence relationships between the operations of the same job expressed in (1b). For any two operations belonging to two distinct jobs and to be performed on the same machine, two disjunctive arcs are included in E: an arc (i, j) with length $d_{ij} = p_i$ and an arc (j, i) in the opposite direction with length $d_{ji} = p_j$. This pair of arcs $\{(i, j), (j, i)\}$ is added to family $\mathcal{E}$ and reflects the corresponding machine capacity constraint in (1c). We sometimes denote a generic pair of disjunctive arcs by $D = \{e, \bar{e}\}$ and refer to $\bar{e}$ as the mate of e. Figure 3 provides the disjunctive graph representation of our illustrative example.

Any feasible solution of (1) describes a fixed operation processing sequence for all machines involved. Intuitively, deciding on the operation processing sequences amounts to choosing exactly one of each pair of disjunctive constraints in (1c) to include in the formulation while excluding the other one. Given the correspondence of (1) to the disjunctive graph representation, choosing disjunctive constraints is generalized and formalized via the notion of "selections" of disjunctive arcs in G and helps us characterize and capture solutions of **JS–CONV**.

Definition 1 *Any subset of disjunctive arcs $S \subseteq E$ is called a selection. A selection S is acyclic if its associated graph $G(S) = (\mathcal{I}, A \cup S, d)$ contains no cycle, and is cyclic otherwise. A selection S is complete if $S \cap D \neq \emptyset$ for all $D \in \mathcal{E}$.*

Given any selection $S \subseteq E$, the solution space of the feasible start times is

$$\Omega(S) = \left\{ \boldsymbol{\alpha} \in \mathbb{Z}^{*|\mathcal{I}|} : \alpha_j - \alpha_i \geq d_{ij} \text{ for all } (i, j) \in A \cup S \right\}.$$

The solution space $\Omega(S)$ is empty if and only if S is cyclic. If S induces a cycle C in $G(S)$, then summing over all constraints $\alpha_j - \alpha_i \geq d_{ij} = p_i$ for $(i, j) \in C$ results in the contradiction $0 \geq \sum_{(i,j)\in C} p_i > 0$ and establishes that $\Omega(S) = \emptyset$. Otherwise, we can always find a set of feasible integer start times. For instance, in order to compute an *earliest start time schedule* a fictive initial node σ and an arc of length 0 from σ to each node $i \in \mathcal{I}$ is introduced into $G(S)$. For each operation $i \in \mathcal{I}$, its *earliest start time* α_i is then the length of a longest path from σ to i.

A feasible schedule of **JS–CONV** requires that we schedule all operations by removing all sequencing conflicts between the operations on the same machine – see (1c). Consequently, a selection S is called *feasible* if S is complete and acyclic. Observe that a feasible selection must pick at least one disjunctive arc from each pair of arcs D in $\mathcal{E}$ by the definition of completeness, and selecting both leads to a cycle. Therefore, in any feasible selection S, exactly one arc is present from every $D \in \mathcal{E}$ – as expected. The solution depicted in Figure 1 corresponds to the feasible selection $\{\overline{e}_1, e_2, e_3, \overline{e}_4, \overline{e}_5, e_6, \overline{e}_7, e_8, e_9, \overline{e}_{10}, \overline{e}_{11}, \overline{e}_{12}\}$ in our illustrative example and is an earliest start time schedule.

If the objective function is regular, no element of $\Omega(S)$ attains a lower scheduling cost for a given acyclic selection S than the earliest start time schedule. However, this schedule is typically not minimizing for the convex cost function stated in our problem. It is easy to see that we can improve upon the earliest start time schedule of Figure 1 by shifting operation 8 forward to start at time 2 and reducing the cost incurred by job O from 14 to 12. The updated feasible schedule is clearly associated with the same selection because the execution order of the operations remains intact. In general, for a given acyclic selection S we need to solve the optimization problem formulated in Section 3 – also known as the timing problem – in order to identify a schedule $\alpha(S) \in \Omega(S)$ with the minimum cost $c(\alpha(S))$, i.e., $c(\alpha(S)) = \min\{c(\boldsymbol{\alpha}) \ : \ \boldsymbol{\alpha} \in \Omega(S)\}$. This viewpoint leads to an alternate combinatorial formulation of **JS–CONV** on the disjunctive graph G: “Among all feasible selections, find a selection S that minimizes $c(\alpha(S))$.” Such a selection is called an *optimal* selection. The next section discusses how to find an optimal schedule $\alpha(S)$ for any acyclic selection S.

3 The timing problem

The timing problem – already introduced conceptually in Sections 1 and 2 – is at the heart of the local search methods developed for job shop scheduling problems. The objective of the timing problem is to determine the optimal operation start times if we are given a fixed operation processing sequence for all machines involved. In this section, we formally introduce the timing problem for **JS–CONV** and show that it is efficiently solvable by drawing upon the modeling and solution approach by Ahuja et al. (2003) for solving the convex cost integer dual network flow problem.

In general, for a given acyclic (and not necessarily complete) selection S and its associated graph $G(S)$ we need to solve the optimization problem formulated below in order to identify a schedule $\alpha(S)$ with the minimum cost in $\Omega(S)$. This is the timing problem associated with **JS–CONV**:

$$c(\alpha(S)) = \text{minimize} \sum_{(i,j)\in Q} f_{ij}(\delta_{ij}) + \sum_{i\in P} b_i(\alpha_i) \tag{2a}$$

$$\text{subject to:}$$

$$\alpha_j - \alpha_i \geq p_i \text{ for all } (i,j) \in A \cup S, \tag{2b}$$

$$(1d) - (1e).$$

For technical reasons that will be evident soon in the sequel, Proposition 1 below contends that the timing problem always admits an optimal solution so that

$$\alpha_i \leq U \text{ for all } i \in \mathcal{I}, \tag{3}$$

where $U = \max_{i\in P}(\lceil \alpha_i^{\min} \rceil) + \sum_{i\in\mathcal{I}} p_i + \sum_{(i,j)\in Q} \lceil \delta_{ij}^{\min} \rceil$, and the sum or maximum over an empty set is zero by convention.

Proposition 1 *There exists an optimal solution of the timing problem* (2) *satisfying constraints* (3).

The proof is relegated to the appendix (see online supplement).

3.1 A transformation into a linear program

The timing problem (2) is an integer optimization problem with a convex objective function and has the same structure as problem (1) in Ahuja et al. (2003). Drawing upon the developments of Ahuja et al. (2003), we transform the timing problem (2) into the form of problem (3) in Ahuja et al. (2003), which is an LP, except that the objective function is piecewise-linear convex.

The timing problem (2) can be transformed into an equivalent LP by assuming – without loss of generality – that each of the convex functions f and b is linear between successive integers, and then representing each linear segment with a new variable and dropping the integrality restrictions (1e). The validity of this approach derives from the fact that each of the piecewise-linear convex functions f and b has integer breakpoints, i.e., its slope changes only at integer values. It follows from this property that there always exists an optimal solution of this LP that is integral (see e.g., Murty, 1976). Consequently, in the development below we assume that each of the convex functions f and b is linear between consecutive integers and ignore the integrality constraints (1e).

We then integrate the precedence constraints (2b) and the time bounds (3) into the objective function. For all $i \in \mathcal{I}$, define

$$B_i(\alpha_i) = \begin{cases} b_i(0) - M\alpha_i & \text{for } \alpha_i < 0, \\ b_i(\alpha_i) & \text{for } 0 \le \alpha_i \le U, \\ b_i(U) + M(\alpha_i - U) & \text{for } \alpha_i > U, \end{cases}$$

where $b_i(\alpha_i) = 0$ for all values of α_i if $i \notin P$, and M is a sufficiently large number. We discuss its possible values below. Similarly, we update the functions f. Let $\delta_{ij} = \alpha_j - \alpha_i$ for all $R(S) = A \cup S \cup Q$. By (3) and the non-negativity of the start times, δ_{ij} is bounded by $-U \le \delta_{ij} \le U$ for all $(i,j) \in R(S)$. In addition, $\delta_{ij} \ge p_i$ must hold for all $(i,j) \in A \cup S$ by the precedence constraints (2b). Consequently, $l_{ij} = p_i$ is a lower bound on δ_{ij} for all $(i,j) \in A \cup S$ and $l_{ij} = -U \le \delta_{ij}$ holds for all $(i,j) \in Q \setminus (A \cup S)$. We also set $f_{ij}(\delta_{ij}) = 0$ for all values of δ_{ij} and for all $(i,j) \in (A \cup S) \setminus Q$. Then, for all $(i,j) \in R(S)$, define

$$F_{ij}(\delta_{ij}) = \begin{cases} f_{ij}(l_{ij}) - M(\delta_{ij} - l_{ij}) & \text{for } \delta_{ij} < l_{ij}, \\ f_{ij}(\delta_{ij}) & \text{for } l_{ij} \le \delta_{ij} \le U, \\ f_{ij}(U) + M(\delta_{ij} - U) & \text{for } \delta_{ij} > U. \end{cases}$$

The functions B and F of our illustrative example are sketched in Figure 4.

The value of M must be chosen such that any $\boldsymbol{\alpha} \ge \mathbf{0}$ that violates some time bound constraint (3) or precedence constraint (2b) cannot be an optimal timing solution, and F_{ij} is convex. To this end, note that f and b are non-negative, let ub be an upper bound on the optimal objective function value of the timing problem (2) – obtained, for instance, from the earliest start time schedule, and compute $z_1 = \max_{i \in P}\{b_i(0) - b_i(1), b_i(U) - b_i(U-1)\}$ and $z_2 = \max_{(i,j) \in Q}\{f_{ij}(l_{ij}) - f_{ij}(l_{ij}+1), f_{ij}(U) - f_{ij}(U-1)\}$. Then, $M \ge \max(ub+1, z_1, z_2)$ fulfills the requirement. In our implementation, we alternatively set M to $1 + \sum_{i \in P} \max(b_i(0), b_i(U)) + \sum_{(i,j) \in Q} \max(f_{ij}(-U), f_{ij}(U))$. This choice of M is valid for all acyclic selections because f and b are non-negative, and the convexity of these functions ensures that their maximum is attained at one of the end points of the related interval.

The transformations above lead to the following LP re-formulation of the timing problem (2), except that the objective is piecewise-linear convex:

$$\text{minimize} \sum_{(i,j) \in R(S)} F_{ij}(\delta_{ij}) + \sum_{i \in \mathcal{I}} B_i(\alpha_i) \tag{4a}$$

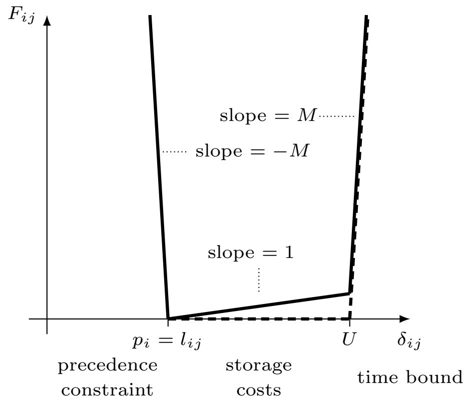
subject to:

$$\alpha_j - \alpha_i = \delta_{ij} \text{ for all } (i,j) \in R(S). \tag{4b}$$

Observe that this problem is equivalent to problem (3) in Ahuja et al. (2003) with a minor difference in the statement of the constraints (4b). Ahuja et al. (2003) consider the difference of $\alpha_i - \alpha_j = \delta_{ij}$, while we prefer $\alpha_j - \alpha_i = \delta_{ij}$ and adapt the results of Ahuja et al. (2003) accordingly.

3.2 A transformation into a network flow problem

As shown by Ahuja et al. (2003), the Lagrangian relaxation of problem (4) obtained by relaxing the constraints (4b) can be transformed into a minimum cost network flow problem with piecewise-linear convex arc cost



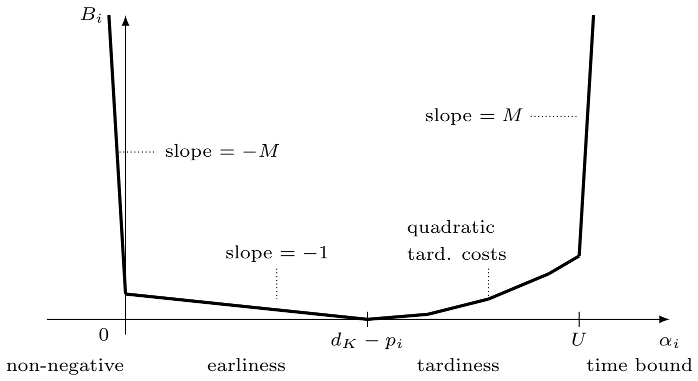


Figure 4: The transformed cost functions in our illustrative example. (left) Function F_{ij} for all $(i,j) \in A$ – see the solid curve. For each selected arc $(i,j) \in S$, the function F_{ij} is identical, except that the storage costs are zero in the interval $[p_i, U]$ – see the dashed curve. (right) Function B_i, where i is the last operation of some job K.

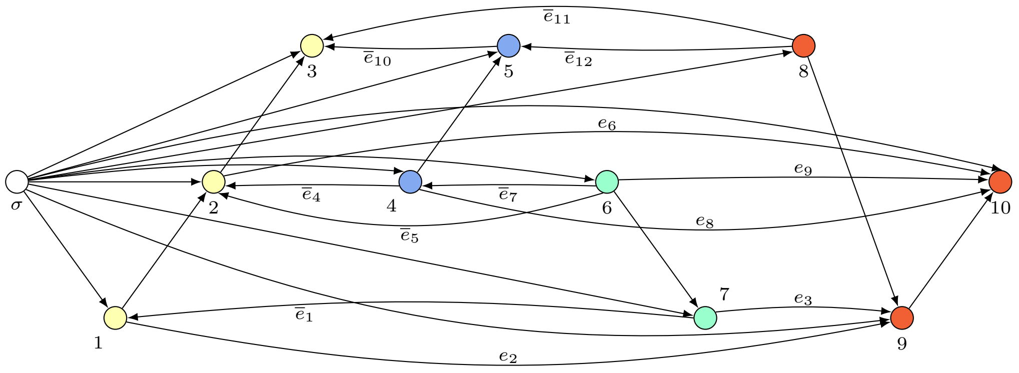


Figure 5: The timing graph $T(S)$ of our illustrative example using selection S associated with the solution in Figure 1.

functions, where each linear segment has an integer slope. We specify here the minimum cost network flow problem in detail and explain how to obtain an optimal timing solution from an optimal flow. For the details and proofs, we refer to Ahuja et al. (2003).

The minimum cost network flow problem is defined on a *timing graph* $T(S) = (\mathcal{I}^+, R(S)^+)$ obtained from the graph $G(S) = (\mathcal{I}, A \cup S, d)$ by adding a source node σ, an arc (σ, i) for each $i \in \mathcal{I}$, and an arc (i,j) for each $(i,j) \in Q \setminus (A \cup S)$. Consequently, the arc set contains one arc for each $(i,j) \in R(S)$ in addition to the arcs (σ, i) for all $i \in \mathcal{I}$ and is labeled as $R(S)^+$. Note that the arc length vector d is no longer needed. Furthermore, in this formulation the operation start times are implicitly represented by the flows out of σ; that is, for each arc $(\sigma, i), i \in \mathcal{I}$, we have $\delta_{\sigma i} = \alpha_i$, $l_{\sigma i} = 0$, and capture the cost $B_i(\alpha_i)$ through $F_{\sigma i}(\delta_{\sigma i})$. Figure 5 depicts the timing graph of our illustrative example given the selection S associated with the solution in Figure 1.

In the resulting minimum cost network flow problem, the flow variable x_{ij} is actually the Lagrangian dual variable associated with the corresponding constraint in (4b):

$$\text{minimize} \sum_{(i,j) \in R(S)^+} C_{ij}(x_{ij}) \tag{5a}$$

subject to:

$$\sum_{j:(i,j) \in R(S)^+} x_{ij} - \sum_{j:(j,i) \in R(S)^+} x_{ji} = 0 \text{ for all } i \in \mathcal{I}^+, \tag{5b}$$

$$-M \leq x_{ij} \leq M \text{ for all } (i,j) \in R(S)^+, \tag{5c}$$

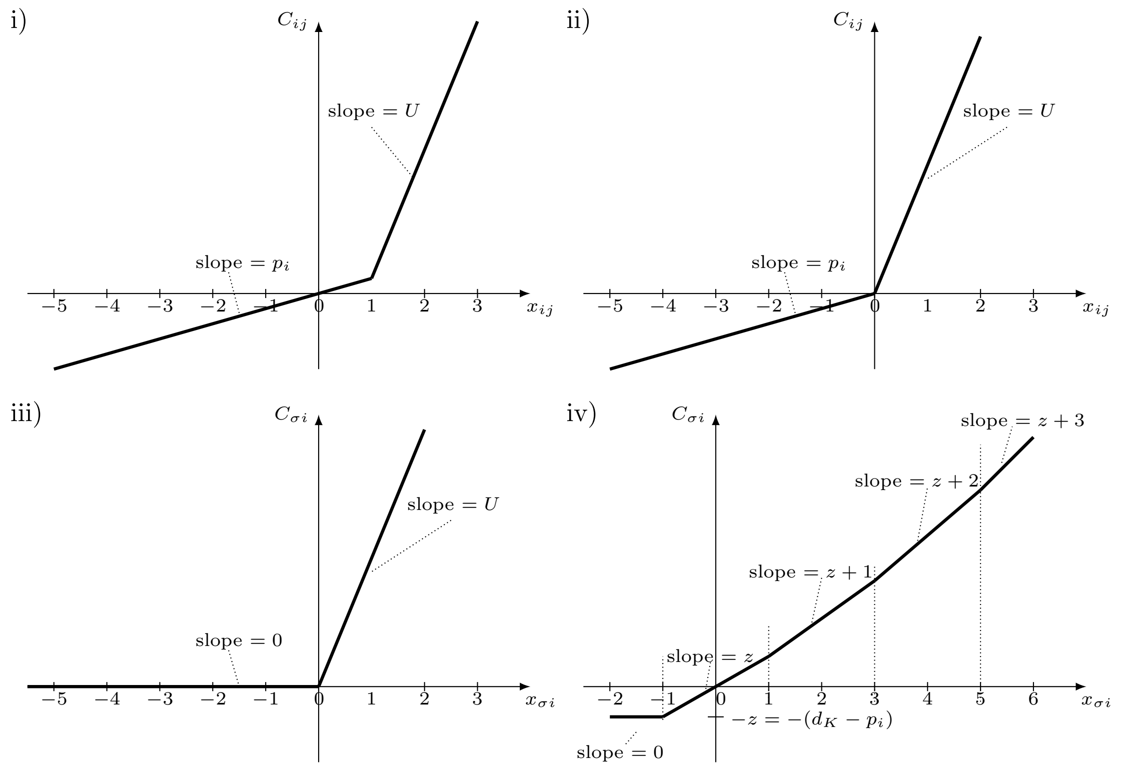


Figure 6: Functions C of our illustrative example: i) C_{ij} for each arc $(i,j) \in A$, ii) C_{ij} for each arc $(i,j) \in S$, iii) C_{ij} for each arc (i,σ), where i is not the last operation of its job, and iv) C_{ij} for each arc arc (i,σ), where i is the last operation of some job K. Note that for i) each arc $(i,j) \in A$ is also present in Q. Indeed, the storage costs modeled with the function f_{ij} are accountable for the difference in shape between the functions of type i) and ii).

where for each arc $(i,j) \in R(S)^+$, the piecewise-linear convex function C_{ij} is described as follows:

$$C_{ij}(x_{ij}) = \begin{cases} -F_{ij}(l_{ij}) + l_{ij}x_{ij} & \text{for } -M \le x_{ij} \le a_{ij}(l_{ij}), \\ -F_{ij}(l_{ij}+1) + (l_{ij}+1)x_{ij} & \text{for } a_{ij}(l_{ij}) \le x_{ij} \le a_{ij}(l_{ij}+1), \\ \dots & \\ -F_{ij}(q) + qx_{ij} & \text{for } a_{ij}(q-1) \le x_{ij} \le a_{ij}(q), \\ \dots & \\ -F_{ij}(U) + Ux_{ij} & \text{for } a_{ij}(U-1) \le x_{ij} \le M. \end{cases} \tag{5d}$$

In these expressions, θ is integer and $a_{ij}(\theta) = F_{ij}(\theta+1) - F_{ij}(\theta)$ for all $l_{ij} \le \theta \le U-1$. The functions C of our illustrative example are sketched in Figure 6.

A remark on the sets S and Q is in order. If S and Q share an element (i,j), it would be convenient to introduce two parallel arcs (i,j) in the timing graph $T(S)$ as it simplifies the discussion of the neighborhood properties developed in the next section. However, parallel arcs substantially increase the notational complexity. In the remainder of this paper, we therefore assume that $E \cap Q = \emptyset$ so that $S \cap Q = \emptyset$ holds for all acyclic selections S. Observe that this assumption is fulfilled for the problems we experiment on in our numerical study in Section 5.

3.3 Solving the network flow and the timing problems

For each arc $(i,j) \in R(S)^+$, the function C_{ij} is convex and piecewise-linear with at most $U - l_{ij} + 1$ linear segments. Therefore, problem (5) could in principle be solved by any minimum cost network flow algorithm in an expanded network containing at most $U - l_{ij} + 1$ arcs for each arc (i,j) in $T(S)$. Because of the huge number of arcs contingent on U, this direct approach would not run in polynomial time in general. We therefore solve problem (5) with the cost-scaling algorithm of Ahuja et al. (2003), which directly works in

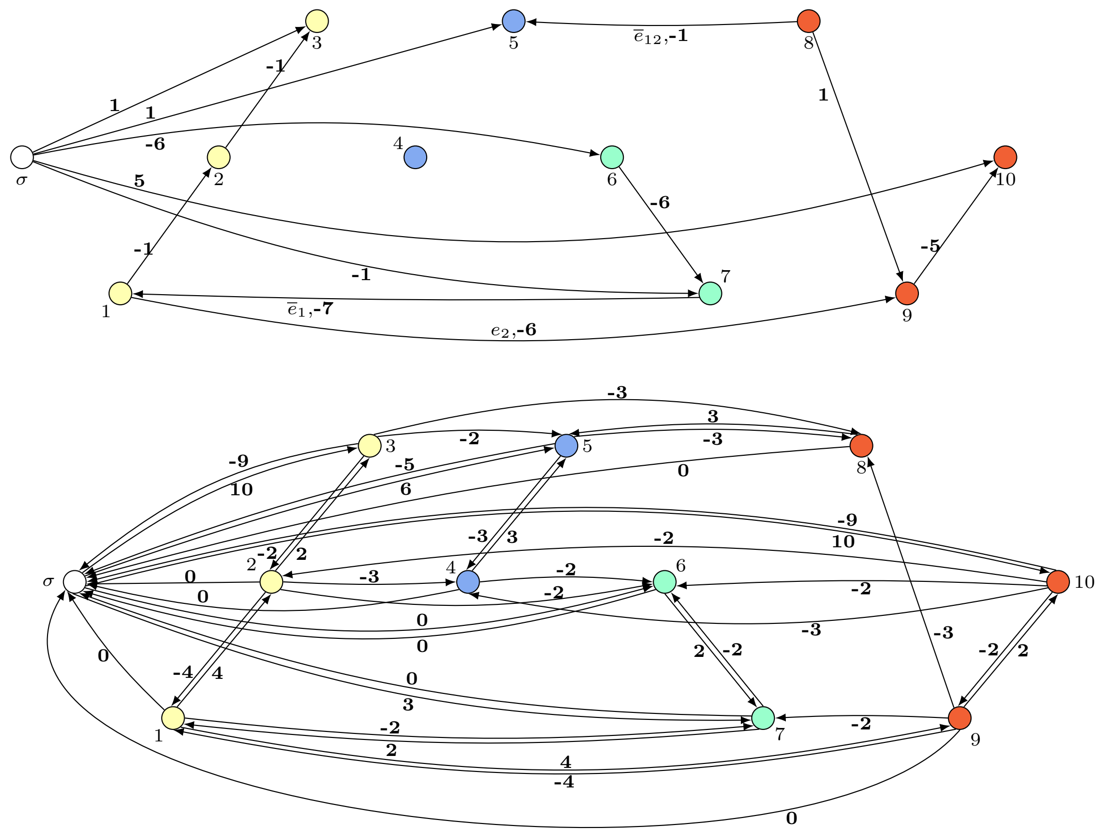


Figure 7: (top) An optimal flow of our illustrative example. All arcs with zero flow are omitted. The arcs $\bar{e}_1, e_2, \bar{e}_{12}$ are critical – see Section 4. (bottom) Residual network $Res(\boldsymbol{x}, S)$ for the optimal flow depicted above ($M = 3579$). All arcs with length $U = 35$ are omitted.

the timing graph $T(S)$. This algorithm runs in polynomial time $\mathcal{O}(qp\log(q^2/p)\log(qU))$, where $q = |I|$ and $p = |R(S)|$. At termination, the algorithm provides an optimal solution of problem (5), i.e., an optimal flow $\boldsymbol{x}$, and an optimal node potential $\boldsymbol{\pi} = \{\pi(i) : i \in I^+\}$.

In order to deduce an optimal solution of the timing problem from these outputs, we introduce a so-called residual network $Res(\boldsymbol{x}, S)$ associated with the timing graph $T(S)$ and flow $\boldsymbol{x}$. For each arc (i, j) in $T(S)$, we add a so-called forward arc (i, j) into $Res(\boldsymbol{x}, S)$ if $x_{ij} < M$ and a so-called backward arc (j, i) if $x_{ij} > -M$. The length of a forward arc (i, j) is set to the right slope of function C_{ij} at point x_{ij}, and the length of a backward arc (j, i) is set to the negative of the left slope of function C_{ij} at point x_{ij}. The left and right slopes of C_{ij} at point x_{ij} are defined as $(C_{ij}(x_{ij}) - C_{ij}(x_{ij} - \delta))/\delta$ and $(C_{ij}(x_{ij} + \delta) - C_{ij}(x_{ij}))/\delta$, respectively, where δ is a sufficiently small number. The reduced cost of an arc (i, j) in $Res(\boldsymbol{x}, S)$ is $c^{\pi}_{ij} = c_{ij} - \pi(i) + \pi(j)$. At the termination of the algorithm, we obtain a pair $(\boldsymbol{x}, \boldsymbol{\pi})$ that satisfies the following optimality conditions:

$$c^{\pi}_{ij} \geq 0 \text{ for all } (i, j) \text{ in } Res(\boldsymbol{x}, S). \tag{6}$$

Figure 7 provides an optimal flow and the corresponding residual network for our illustrative example.

The node potential $\boldsymbol{\pi}$ may have fractional components, and $\pi(\sigma)$ may be different from zero. However, the piecewise-linear structure of the objective with integral breakpoints assures us of the existence of an integral optimal node potential $\boldsymbol{\pi}'$ with $\pi'(\sigma) = 0$. A node potential $\boldsymbol{\pi}'$ with these attributes can be determined by computing the length ℓ_i of a shortest path from $i \in \mathcal{I}$ to σ in the residual network $Res(\boldsymbol{x}, S)$ and setting $\pi'(i)$ to ℓ_i. The optimal timing solution $\alpha(S)$ is finally $\alpha_i(S) = -\pi'(i)$ for all $i \in \mathcal{I}$. In our illustrative example, we get the optimal solution depicted in Figure 8.

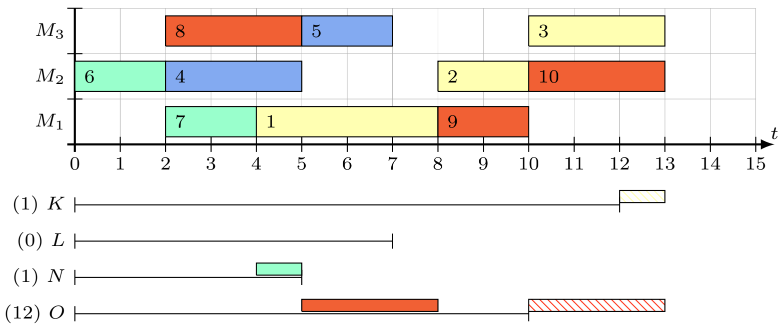


Figure 8: An optimal timing solution of our illustrative example with cost 14.

4 A local search heuristic for JS–CONV

In principle, the previous sections provide an exact approach for finding an optimal selection: Enumerate all feasible selections, evaluate their costs using the network flow-based approach, and return a selection (and its optimal schedule) with minimum optimal timing cost. As is well-known in job shop scheduling, this procedure is too time consuming even for small instances. In order to find a good selection within a reasonable time, we propose a local search approach. Such methods have been applied with great success to various job shop scheduling problems in the past – see Section 1. In this section, we first introduce an operation-swap-based neighborhood and then describe a tabu search.

4.1 An operation-swap-based neighborhood for JS–CONV

The underlying pillar of any local search is a neighborhood function N that associates a set of neighbor selections $N(S)$ with each feasible selection S. Many well-known neighborhoods for job shop scheduling problems are based on swapping two critical consecutive operations on a machine. This concept was used, for example, by Balas (1969) in the classical job shop, by De Bontridder (2005) in the job shop total weighted tardiness problem, by Bürgy (2017) in a job shop problem with a general regular objective, and by Wennink (1995); Brandimarte and Maiocco (1999); Avci and Storer (2004) in job shop problems with non-regular objectives.

By following a framework common in this body of literature, we define a neighborhood for **JS–CONV**. More specifically, given any feasible selection S together with an optimal flow $\boldsymbol{x}$, an optimal integer potential $\boldsymbol{\pi}$ with $\pi(\sigma) = 0$, and an optimal timing solution $\alpha(S)$ with $\alpha_i(S) = -\pi(i)$ for all $i \in \mathcal{I}$, we first define the notion of a critical arc, and then derive a neighborhood based on replacing critical arcs. Finally, we establish that all solutions in the neighborhood are feasible and prove the opt-connectedness of the neighborhood.

Definition 2 *An arc $(i, j) \in S$ is labeled as critical if $x_{ij} < 0$, and non-critical otherwise. Let $S^{crit} = \{e \in S : x_e < 0\}$ be the set of all critical arcs.*

Proposition 2 *For any critical arc (i, j), $\alpha_j(S) - \alpha_i(S) = p_i$ holds.*

Proof. As $\alpha(S)$ are feasible start times, $\alpha_j(S) - \alpha_i(S) \geq p_i$ holds by (2b), and it suffices to show that $\alpha_j(S) - \alpha_i(S) \leq p_i$ given that $x_{ij} < 0$ as required by Definition 2 for a critical arc. Consider the residual network $Res(\boldsymbol{x}, S)$, and note that $(i, j) \notin Q$ because we assumed $S \cap Q = \emptyset$. Therefore, F_{ij} has exactly the shape of the dotted curve on the left in Figure 4, and the right slope of the resulting cost function C_{ij} at $x_{ij} < 0$ is $l_{ij} = p_i$ – see ii) in Figure 6. Consequently, arc (i, j) is present in $Res(\boldsymbol{x}, S)$ with a length of p_i because $x_{ij} < M$. The optimality of $(\boldsymbol{x}, \boldsymbol{\pi})$ requires $c^{\pi}_{ij} = p_i - \pi(i) + \pi(j) \geq 0$ as stated in (6), or equivalently, $p_i + \alpha_i(S) - \alpha_j(S) \geq 0$, and the result follows. □

In other words, if arc (i, j) is critical, i and j are processed consecutively on the same machine without any idle time in between. Clearly, this is a generalization of the well-known concept of criticality based on the longest path calculations in $G(S)$ if the objective function is regular. The number of critical arcs is no more than $|\mathcal{I}| - |\mathcal{M}|$ because each operation has at most one immediate successor operation on the same machine, and the final operation on a machine has none. It turns out that all non-critical arcs carry zero flow as formalized in the next lemma. The proof is in the appendix (see online supplement).

Lemma 1 *For all $e \in S \setminus S^{crit}$, $x_e = 0$ holds.*

This, in fact, paves the way to showing that the critical arcs totally determine the cost of selection S, which is formally stated as:

Theorem 1 *$c(\alpha(S^{crit})) = c(\alpha(S))$ holds.*

Proof. All arcs $e \in S \setminus S^{\text{crit}}$ have $x_e = 0$ by Lemma 1. Therefore, flow $\boldsymbol{x}$ is also feasible with respect to the timing graph $T(S^{\text{crit}})$ because the set of arcs in $T(S^{\text{crit}})$ is only missing the arcs $S \setminus S^{\text{crit}}$ with zero flow in $T(S)$. That is, we have $R(S^{\text{crit}})^+ \subseteq R(S)^+$ as $S^{\text{crit}} \subseteq S$, and the arcs of $Res(\boldsymbol{x}, S^{\text{crit}})$ are all contained in $Res(\boldsymbol{x}, S)$ with the same length. Consequently, the pair $(\boldsymbol{x}, \boldsymbol{\pi})$ fulfills the optimality conditions (6) with respect to $Res(\boldsymbol{x}, S^{\text{crit}})$ and certifies $\alpha(S)$ as an optimal timing solution for S^{crit} with the same cost. □

The fundamental implication of Theorem 1 is that we must replace at least one critical arc in S in order to be able to identify an improved selection S' with $c(\alpha(S')) < c(\alpha(S))$. This justifies the following neighborhood definition.

Definition 3 *For each critical arc $e = (i, j)$, a neighbor $S_e = S \cup \overline{e} \setminus e$ is constructed by removing e from S and adding $\overline{e} = (j, i)$. The neighborhood $N(S)$ of S consists of all of these neighbors.*

As the number of critical arcs is at most $|\mathcal{I}| - |\mathcal{M}|$, the cardinality of $N(S)$ is bounded from above by $|\mathcal{I}| - |\mathcal{M}|$. To illustrate the neighborhood generation mechanism applied to our illustrative example, observe that the optimal timing solution presented in Figure 8 corresponds to the selection $S = \{\overline{e}_1, e_2, e_3, \overline{e}_4, \overline{e}_5, e_6, \overline{e}_7, e_8, e_9, \overline{e}_{10}, \overline{e}_{11}, \overline{e}_{12}\}$. Inspecting the associated optimal flow $\boldsymbol{x}$ in Figure 7 (top) reveals $S^{\text{crit}} = \{\overline{e}_1, e_2, \overline{e}_{12}\}$ as the set of critical arcs. Thus, the neighborhood of S is determined as $N(S) = \{S_{\overline{e}_1}, S_{e_2}, S_{\overline{e}_{12}}\}$. The optimal timing solutions for these three neighbors are provided in Figure 9.

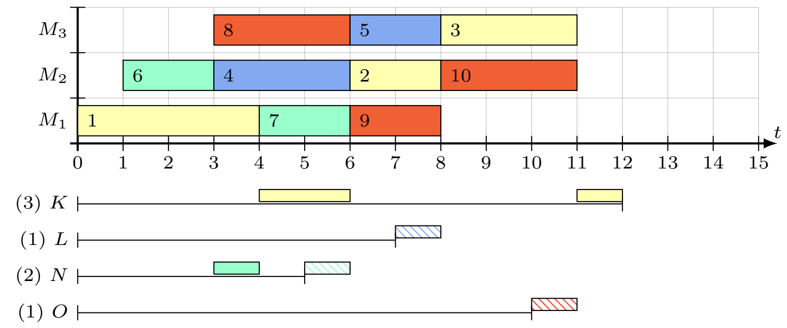
The time expended for neighborhood evaluations is typically the computational bottleneck in local search methods. In this context, two issues require close attention. First, substituting a critical arc by its mate is necessary but not sufficient for an improving move – the second and third neighbors in Figure 9 lead to substantially worse objective values compared to the current solution in Figure 8, and this is a common phenomenon in local search methods for job shop scheduling problems. Therefore, a lot of effort has been put into compact neighborhood definitions with the goal of reducing the neighborhood size. For instance, a classical result for $Jm//C_{\max}$ (and other regular objectives) is due to Matsuo et al. (1988), who recognized that given a maximal set of consecutive critical arcs between operations performed on the same machine – a so-called critical machine block – an improving move must involve at least one of the first or last critical arcs in the block. This result is then leveraged to drop the *internal* critical arcs from the neighborhood. However, we observed that swapping such an arc may actually decrease the objective function value for **JS–CONV** and identify the development of a compact neighborhood definition as a future research direction in Section 6. Second, a closely related matter is ruling out infeasible solutions from the neighborhood. The following theorem establishes that all selections in $N(S)$ are feasible.

Theorem 2 *For any $S' \in N(S)$, S' is feasible.*

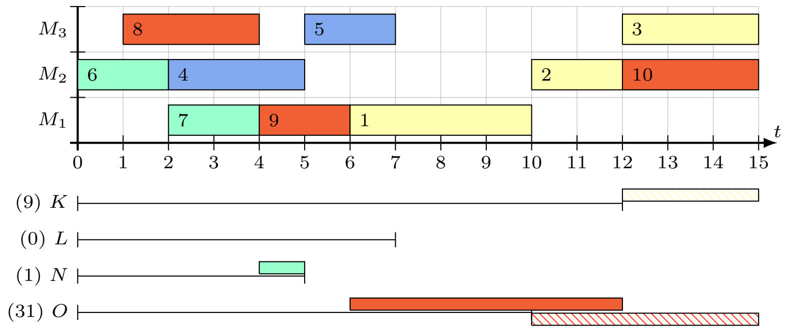
Proof. The proof is similar to that of Proposition 2 in Balas (1969). By Definition 3, S' is generated from S by replacing some critical arc $e = (i, j)$ so that $S' = S \cup \overline{e} \setminus e$. As S is feasible, i.e., complete and acyclic, S' is complete. It remains to show that S' is acyclic.

Suppose that e is critical and S' is cyclic. Then, graph $G(S')$ contains some cycle C. However, selection $S' \setminus \overline{e} \subset S$ is acyclic, and graph $G(S' \setminus \overline{e})$ does not contain any cycle. Therefore, C must include arc $\overline{e} = (j, i)$, and a path P_{ij} starting from $v_0 = i$, going through the nodes $v_1, \ldots, v_n$, and terminating at $v_{n+1} = j$.

Neighbor $S_{\bar{e}_1}$ with total cost 7 obtained by moving operation 1 before operation 7:



Neighbor S_{e_2} with total cost 41 obtained by moving operation 9 before operation 1:



Neighbor $S_{\bar{e}_{12}}$ with total cost 27 obtained by moving operation 5 before operation 8:

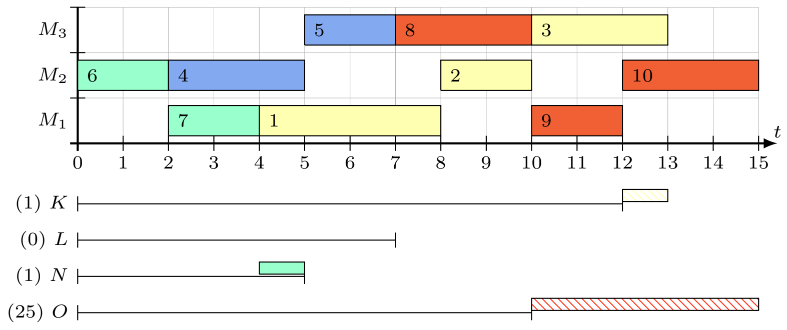


Figure 9: The three neighbors of the solution depicted in Figure 8.

Clearly, P_{ij} is also present in $G(S)$ and can contain no less than two arcs because $e = (i,j)$ is not part of $G(S')$; that is, $n \geq 1$. By (2b), any feasible timing solution $\boldsymbol{\alpha} \in \Omega(S)$ must satisfy $\alpha_{v_r} - \alpha_{v_{r-1}} \geq p_{v_r}$ for all $r = 1, ..., n+1$. Summing up over these inequalities yields $\alpha_j - \alpha_i \geq p_i + \sum_{r=1}^{n} p_{v_r}$. As $n \geq 1$ and $p_{v_r} > 0$ for all $v_r \in \mathcal{I}$, $\sum_{r=1}^{n} p_{v_r} > 0$. This implies that $\alpha_j - \alpha_i > p_i$ and is in contradiction with Proposition 2 given that (i,j) is critical. $\square$

Finally, we prove that our neighborhood is opt-connected. This is a desirable characteristic for a neighborhood definition, but is of little practical relevance from a computational viewpoint.

Theorem 3 *If S is not optimal then there exists a finite sequence of selections $S_1, S_2, ..., S_n$ starting with $S_1 = S$ and terminating with an optimal selection $S^* = S_n$ such that $S_i \in N(S_{i-1})$ for all $i = 2, ..., n$.*

Proof. The proof is similar to that of Theorem 1 in Van Laarhoven et al. (1992). Given an optimal selection S_{opt}, let $\Delta(S)$ and $\Delta^{\text{crit}}(S)$ represent the number of arcs of S and S^{crit} not present in S_{opt}, respectively. That is, $\Delta(S) = |S \setminus S_{\text{opt}}|$ and $\Delta^{\text{crit}}(S) = |S^{\text{crit}} \setminus S_{\text{opt}}|$. We distinguish between two cases:

a) $\Delta^{\text{crit}}(S) = 0$. Then, $S^{\text{crit}} \subseteq S_{\text{opt}}$, and the relation $c(\alpha(S^{\text{crit}})) \leq c(\alpha(S_{\text{opt}})$ is satisfied by the definition of the timing problem (2). We conclude that S is an optimal selection because Theorem 1 implies $c(\alpha(S)) = c(\alpha(S^{\text{crit}})) \leq c(\alpha(S_{\text{opt}})$.

b) $\Delta^{\text{crit}}(S) > 0$. Let $e \in S^{\text{crit}} \setminus S_{\text{opt}}$. By Definition 3, $S_e = S \cup \overline{e} \setminus e$ is in $N(S)$. The completeness of S^{opt} yields $\overline{e} \in S^{\text{opt}}$ and $\Delta(S_e) = \Delta(S) - 1$. Equivalently, moving from S to S_e reduces the number of arcs of the current selection not present in S^{opt} by one. Thus, we arrive at the conclusion that in at most $\Delta(S)$ steps, S can be transformed into a selection S^* with $\Delta^{\text{crit}}(S^*) = 0$, which is optimal by a). $\square$

4.2 A tabu search for JS–CONV

Tabu search is a prominent local search approach in combinatorial optimization (see e.g., Glover and Laguna, 1997; Glover, 1996). In particular, it has been successfully applied for solving various job shop scheduling problems, such as, for example, the classical job shop problem (Nowicki and Smutnicki, 1996; Peng et al., 2015), job shop problems with routing flexibility (Brandimarte, 1993; Kis, 2003), and more complex job shop scheduling problems (Hurink and Knust, 2005; Drótos et al., 2009; Bürgy, 2017). We present a simple tabu search for **JS–CONV** using the neighborhood developed in the previous section. In what follows, we discuss the construction of the tabu list, the evaluation of the neighborhood, and the stopping criterion. We refer to Glover and Laguna (1997) for a detailed treatment of tabu search.

A tabu list L storing the disjunctive arcs that have been replaced during the last $maxL$ iterations is maintained. List L is empty at the beginning of the search. Upon the completion of a move from a selection S to S_e by replacing arc $e \in S$ through its mate $\overline{e}$, we insert e into L at the head position, and the last entry in the list is deleted if the size of L exceeds $maxL$. A neighbor S' is called tabu if $S' \cap L \neq \emptyset$. In our numerical experiments, $maxL$ is set to 16.

The evaluation of a selection S consists of determining an optimal schedule $\alpha(S)$ by the approach developed in Section 3. Initially, the objective value of S is set to $c(\alpha(S))$. However, if S is tabu and does not improve upon the best selection identified so far, a penalty term $(maxL - k)M$ is added to the objective of S, where k indicates the position in L of the first disjunctive arc in $L \cap S$, and M is a large constant – its value may be determined as specified in Section 3. Among all evaluated neighbors, we move to a neighbor with the lowest objective value. As the neighborhood size is typically quite large and the evaluation of one neighbor is expensive – see Table 1 in Section 5.1, we randomly pick and evaluate at most $maxS$ neighbors of the current selection. In our numerical experiments, $maxS$ is set to 15.

The tabu search is stopped after reaching a pre-specified time limit $maxT$ or if the neighborhood of the current selection S is empty, which certifies S as optimal by Theorem 3. However, this case was never realized in our tests. The parameter $maxT$ is set to 1800 s in our numerical experiments.

5 An application: just-in-time job shop scheduling

In order to demonstrate that the proposed tabu search (TS)-based solution approach is able to find high-quality solutions within reasonable computation times, we apply it to two just-in-time job shop scheduling problems and conduct extensive computational tests.

Just-in time scheduling problems are characterized by the goal of scheduling the operations at the right moment rather than as early as possible (Monette et al., 2009). Earliness is typically related to increasing storage costs and the deterioration of intermediate or finished goods, and tardiness may result in customer dissatisfaction, loss of reputation, and lost sales (Parsa et al., 2017). In job shop scheduling, just-in-time objectives are typically modeled by introducing a fixed due date for each job and penalizing the intermediate storage as well as the early and tardy completion of the jobs.

We employ our approach for the following two types of objectives. First, we consider linear storage, earliness, and tardiness costs – see Section 5.2. We denote this problem by JIT-JS-LIN. This is the objective typically incorporated in the literature (Sadeh, 1996; Brandimarte and Maiocco, 1999; Avci and Storer, 2004; Bülbül and Kaminsky, 2013), and we compare our approach to the most recent results of Bülbül and Kaminsky (2013). Second, in order to test the ability of our approach to address non-linear objectives, we substitute superlinear tardiness costs for linear tardiness costs – the storage and earliness costs are still linear. This problem is denoted by JIT-JS-SUPERLIN. In Section 5.3, we present, to the best of our knowledge, the first extensive computational study of a just-in-time job shop scheduling problem with a non-linear objective.

Outstanding among the accomplishments of this research is that both JIT-JS-LIN with a non-regular linear scheduling objective and JT-JS-SUPERLIN with a non-regular and non-linear scheduling objective are handled successfully by the exact same solution approach. We clearly demonstrate that TS represents the new state-of-the-art for JIT-JS-LIN by beating the previous best algorithm for this problem by Bülbül and Kaminsky (2013). For JIT-JS-SUPERLIN, the only available benchmark is a mixed integer linear programming (MIP) formulation solved by an off-the-shelf solver, and we report solutions of superior quality against the best available feasible solutions from MIP. Collectively, the results in this section are a testament to our claim to generality in Section 1. A second major conclusion from our computational study is that TS delivers these results despite a deliberately simple design, which – for instance – lacks an elaborate diversification mechanism. Thus, the favorable numerical results can actually be attributed to the quality of the neighborhood definition based on the timing solution in Section 4 and justifies the focus of our paper on the timing problem in Section 3. In the following subsections, we specify the experimental setting and discuss the obtained results in detail.

5.1 Experimental settings

In problem JIT-JS-LIN, a release time r_K and a due date d_K are present for each job $K \in \mathcal{J}$. The first operation $i = K_1$ of K cannot start earlier than r_K and a storage cost of c_K^{stor} per time unit is incurred for the time between r_K and the start of i. This can be modeled in **JS–CONV** by specifying the convex function $b_i(\alpha_i) = \max\{0, (\alpha_i - r_K)c_K^{\text{stor}}\}$ and setting the lower bound $l_{\sigma i}$ of α_i to r_K (or placing sufficiently high costs for all $\alpha_i < r_K$). Furthermore, for each pair of consecutive operations i and j in K, an intermediate storage cost of c_{ij}^{storInt} per time unit is charged against the time between the end of i and the start of j. This can be modeled in **JS–CONV** by specifying the function $f_{ij}(\delta_{ij}) = (\delta_{ij} - p_i)c_{ij}^{\text{storInt}}$. Finally, if K is finished earlier than its due date d_K, the time between the completion time of K and d_k – the earliness of K – is penalized at a rate of c_K^{early} per unit time. Otherwise, the completion of K past d_K leads to a tardiness cost of c_K^{tardy} per unit time. These two cost components associated with the final operation $i = K_{|K|}$ are reflected by the function $b_i(\alpha_i) = \max\left\{(d_K - p_i - \alpha_i)c_K^{\text{early}}, (\alpha_i - d_K + p_i)c_K^{\text{tardy}}\right\}$ in **JS–CONV**. We executed tests with this objective type on a subset of the instances generated by Bülbül and Kaminsky (2013), which includes small instances with 40 operations (10 jobs and 4 machines) and larger instances with 100 operations (10 jobs and 10 machines).

In problem JIT-JS-SUPERLIN, each job K has a due date d_K as in JIT-JS-LIN; however, the release time is 0 for all jobs, and we do not penalize the storage between the release time and the start of the job's

Table 1: Details on the TS iterations for the JIT-JS-SUPERLIN instances. For each instance size $n \times m$ in column one, the average neighborhood size (nb size), the average computation time per neighbor evaluation (time/nb) in milliseconds (ms), and the average computation time per TS iteration (time/iteration) in ms are displayed in columns 2-4, respectively, with the number of neighbors evaluated restricted to $maxS = 15$. All values are rounded to the nearest integer.

inst. size	nb size	time/nb	time/iteration
10×5	18	5	74
15×5	33	10	156
20×5	54	15	248
10×10	21	13	212
15×10	40	27	445
20×10	58	46	757
30×10	111	101	1729
15×15	41	54	873

first operation. This is similar to the setting in Brandimarte and Maiocco (1999); Ohta and Nakatanieng (2006) and reflects situations, where resources are not committed to a job before the start of the job's first operation, and no storage costs are therefore accumulated. The unit earliness cost c_K^{early} for job K and the unit intermediate storage costs c_{ij}^{storInt} for all consecutive operations i and j are set to 1. Finally, the tardiness costs are calculated by taking the tardiness to the power of l. That is, job K is charged a tardiness cost of $b_i(\alpha_i)(\alpha_i - d_K + p_i)^l$, where $i = K_{|K|}$. In our experiments, we set l to 1.3.

The following reasons justify the choice of this objective. First, the earliness costs model situations were early delivery is not possible (see e.g., Brandimarte and Maiocco, 1999). Thus, the storage and earliness costs here measure the total idle time for each job, which is equal to the total time a job spends in the system (i.e., its flow time) minus the job's total processing time. This performance measure can be used to keep the total number of jobs in the system (i.e., the work-in-process) at a low level. Second, the superlinear tardiness costs are particularly interesting if the marginal tardiness cost of a job increases with its tardiness (Gonçalves et al., 2016) – a typical feature in practice (Hoitomt et al., 1990). Preliminary tests revealed that quadratic tardiness costs corresponding to $l = 2$ are quite aggressive so that the storage and earliness costs have almost no impact if it is not possible to schedule all jobs within their due dates. The chosen factor $l = 1.3$ is therefore better suited for our experiments. Third, the simple structure of the objective makes it possible to specify all parameter values here. Hence, other researchers can easily apply this objective for their experiments.

We combined the proposed objective with the standard benchmark set by Lawrence (1984) frequently used for $Jm//C_{\max}$. The Lawrence instances la01 to la40 include 50 to 300 operations (10 to 30 jobs and 5 to 15 machines). The due dates were calculated with the rule of Eilon and Hodgson (1967), i.e., $d_K = \lfloor f \cdot \sum_{i \in K} p_i \rfloor$ for job K, where f is referred to as the due date tightness factor. It is well-known that, if the due dates are tight and tardiness is penalized in a superlinear fashion, tardiness costs dominate the total cost and there is little sense in considering a non-regular objective (Brandimarte and Maiocco, 1999). In contrast, if the due dates are very loose and the tardiness costs are high compared to the other cost components, it is typically worthwhile to consider the due dates as hard deadlines and solve a mirrored version of the problem without any tardiness costs in order to arrive at a high-quality solution for the original problem (see e.g., Ahmadi and Bagchi, 1992). Both cases avoid the need to trade off tardiness costs versus the other cost components and may yield simpler problems from a practical point of view. Consequently, in an effort to demonstrate the value of our approach we tried to set the f-values so that the tardiness costs are well balanced against the storage and earliness costs in good solutions (see the last column in Tables 2-5).

A MIP model derived from the disjunctive programming formulation (1) and solved for each instance via the solver `Gurobi 6.5` on four threads with a time limit of 3600 s serves as a benchmark against TS for both problem types JIT-JS-LIN and JIT-JS-SUPERLIN. The MIP model is available in the appendix (see online supplement). The shifting bottleneck heuristic of Bülbül and Kaminsky (2013) is the second basis of comparison for JIT-JS-LIN.

TS was implemented in Java and run on a PC with a 3.3 GHz Intel® Core$^{\text{TM}}$ i5-4590 processor and 16 GB memory. In order to assess the stochastic behavior of TS, we executed ten independent runs for every instance, each with a time limit of $maxT = 1800$ s. To put this time limit into perspective, consider Table

1. The number of neighbors of the current selection S may be quite high, especially for the large instances, and precludes us from evaluating the entire neighborhood $N(S)$. Even with the current scheme of evaluating no more than $maxS = 15$ neighbors in one TS iteration, an iteration of TS is computationally expensive as evident from column four of Table 1 – the times per iteration range from 74 ms for instances of size 10×5 to nearly 2 s for instances of size 30×10. These times translate to only about 25,000 and 1,000 iterations for instances of size 10×5 and 30×10 within the time limit, respectively. For a local search with a swap-based neighborhood, these iteration numbers are very low. Moreover, we observe another striking fact by comparing the figures in columns three and four in Table 1. About 90% of the computation time in TS is spent toward evaluating the neighbors, given that the total time needed for the neighbor evaluations in one TS iteration is roughly $maxS$ times the time expended for evaluating a single neighbor. TS runs on a single thread in the current implementation, and the neighborhood evaluation by solving timing problems would scale linearly with the number of threads in a parallel implementation. The computational results in the next sections should be evaluated with this background information in mind; in principle, the same solution quality can be attained in a much shorter amount of time. A parallel implementation and developing a faster solution algorithm for the timing problem are both on our future research agenda.

For both JIT-JS-LIN and JIT-JS-SUPERLIN, an initial solution for TS is generated by the solution approach proposed in Bürgy (2017) after creating a corresponding instance with a regular objective by ignoring the storage and earliness costs. The time allotted to this method is 60 s. Given the currently expensive nature of the TS iterations, the quality of the initial solution may factor into the final solution quality in a significant way. This is also investigated in the sequel.

5.2 JIT-JS-LIN

5.2.1 Small JIT-JS-LIN instances

We first analyze the results obtained for the small JIT-JS-LIN instances with 4 machines and 10 jobs. Table 2 provides the results obtained with the MIP approach, the benchmark results (BK) of Bülbül and Kaminsky (2013), and the average initial value, the average final value and the best final value of the ten TS runs. In order to compare TS with MIP and BK, we compute the relative percent deviations (RPD) of the average and best TS results from the benchmark values of MIP and BK. For a pair $res - bench$, RPD is expressed as $100 \cdot (res - bench)/bench$ %, where res and $bench$ refer to the result from TS and the benchmark value, respectively.

The MIP approach is able to solve all instances to optimality within 300 seconds. TS exhibits an outstanding solution quality by finding near-optimal results for all instances. In all runs, the final objective value is within 5% of the optimum, and RPD is already smaller than 1% on average at 300 s – refer to Figure 10 as explained below. Moreover, an optimal solution is identified in 36 out of 40 instances in at least one of the ten runs. TS also compares favorably with BK. In particular, TS provides substantially better results than BK for instances with $f = 1.3, 1.5$ and slightly better results for instances with $f = 1.7$. The small difference of the RPDs associated with the average and the best objective values obtained from TS attests to the robustness of the solutions generated by TS.

In order to analyze the progress of TS over time, we recorded the objective values of the initial solution and the best solution found so far after 100, 300, 600, and 1800 seconds for each run and computed the associated RPDs with respect to the optimal objective function value. We consider the RPDs of all runs of all instances with the same f-value together and calculate the minimum, first quartile, median, third quartile, and maximum RPD. Together with the average RPDs indicated by diamonds, Figure 10 provides these values in box plots. For $f = 1.3$, the total tardiness cost contributes, on average, about 60% to the overall cost (see the last column in Table 2), which implies that the due dates are quite tight. Therefore it is not surprising that the quality of the initial solutions are good. Indeed, the associated optimality gap is about 1.4% on average. TS is then able to find optimal or near-optimal solutions within the first minutes of the search for most of the instances. As expected, the initial solutions have a slightly higher optimality gap for the instances with $f = 1.5$ and 1.7, but they are still of remarkable quality. Also for these instances, TS substantially reduces the optimality gaps in the first minutes of the search and quickly finds near-optimal

Table 2: Detailed results for the small JIT-JS-LIN instances of Bülbül and Kaminsky (2013) with $f = 1.3$ (top), $f = 1.5$ (middle), and $f = 1.7$ (bottom). Column one denotes the instance, column two gives the optimal value (opt) obtained with the MIP formulation, column three presents the reference value (BK) of Bülbül and Kaminsky (2013), and the columns four through six depict the average initial value (init), the average final value (avg), and the best final value (best) of the ten TS runs, respectively. The next four columns compare these values by displaying the relative percent deviation (RPD) of avg from opt, best from opt, avg from BK, and best from BK, respectively. For each instance, the percentage of the total cost arising from the tardiness of the jobs in a best TS solution is exhibited in the final column. All values rounded to one decimal place.

$f = 1.3$

	MIP	BK	TS			RPD − opt		RPD − BK		tard.
inst	opt	value	init	avg	best	avg	best	avg	best	%
Jm_01	3133.2	3456.5	3194.7	3176.7	3133.2	1.4	0.0	-8.1	-9.4	66.9
Jm_02	3068.3	3568.2	3089.0	3077.5	3068.3	0.3	0.0	-13.8	-14.0	61.4
Jm_03	2045.4	2199.1	2094.9	2045.7	2045.4	0.0	0.0	-7.0	-7.0	67.5
Jm_04	1677.8	1876.8	1682.1	1678.4	1677.8	0.0	0.0	-10.6	-10.6	60.0
Jm_05	3351.0	3427.7	3399.8	3352.6	3351.0	0.0	0.0	-2.2	-2.2	66.0
Jm_06	2915.9	3147.7	3047.4	3026.2	2990.2	3.8	2.5	-3.9	-5.0	61.6
Jm_07	2072.0	2072.0	2072.0	2072.0	2072.0	0.0	0.0	0.0	0.0	55.4
Jm_08	1765.1	1765.1	1765.1	1765.1	1765.1	0.0	0.0	0.0	0.0	56.6
Jm_09	2157.9	2241.0	2161.7	2159.4	2157.9	0.1	0.0	-3.6	-3.7	65.0
Jm_10	2119.2	2119.2	2165.7	2124.5	2119.2	0.2	0.0	0.2	0.0	64.8
					average	0.6	0.3	-4.9	-5.2	62.5

$f = 1.5$

	MIP	BK	TS			RPD − opt		RPD − BK		tard.
inst	opt	value	init	avg	best	avg	best	avg	best	%
Jm_01	2474.0	2567.7	2544.3	2507.4	2494.1	1.3	0.8	-2.3	-2.9	56.7
Jm_02	2487.3	2751.9	2499.8	2491.4	2487.3	0.2	0.0	-9.5	-9.6	49.3
Jm_03	1651.8	1702.0	1716.0	1671.3	1651.8	1.2	0.0	-1.8	-2.9	51.1
Jm_04	1393.8	1400.1	1420.9	1395.7	1393.8	0.1	0.0	-0.3	-0.4	44.6
Jm_05	2844.3	2844.3	2919.6	2844.4	2844.3	0.0	0.0	0.0	0.0	54.5
Jm_06	2240.5	2584.5	2258.4	2242.9	2240.5	0.1	0.0	-13.2	-13.3	47.2
Jm_07	2193.1	2266.4	2243.6	2208.0	2201.5	0.7	0.4	-2.6	-2.9	45.7
Jm_08	1430.9	1470.1	1430.9	1430.9	1430.9	0.0	0.0	-2.7	-2.7	32.8
Jm_09	1774.3	1923.3	1785.5	1776.2	1774.3	0.1	0.0	-7.6	-7.7	53.2
Jm_10	1900.4	1973.2	1976.7	1904.4	1900.4	0.2	0.0	-3.5	-3.7	54.0
					average	0.4	0.1	-4.4	-4.6	48.9

$f = 1.7$

	MIP	BK	TS			RPD − opt		RPD − BK		tard.
inst	opt	value	init	avg	best	avg	best	avg	best	%
Jm_01	2038.0	2065.4	2078.1	2060.1	2038.0	1.1	0.0	-0.3	-1.3	40.3
Jm_02	2210.7	2223.2	2272.1	2212.7	2210.7	0.1	0.0	-0.5	-0.6	35.2
Jm_03	1482.1	1482.1	1526.2	1486.6	1482.1	0.3	0.0	0.3	0.0	33.7
Jm_04	1196.6	1197.0	1278.6	1197.0	1196.6	0.0	0.0	0.0	0.0	28.1
Jm_05	2578.5	2578.5	2767.2	2578.5	2578.5	0.0	0.0	0.0	0.0	43.2
Jm_06	1998.7	2036.8	2017.3	2003.2	1998.7	0.2	0.0	-1.7	-1.9	26.9
Jm_07	1980.1	1993.3	2331.1	1983.1	1980.1	0.1	0.0	-0.5	-0.7	33.3
Jm_08	1571.0	1696.6	1590.8	1573.4	1571.0	0.2	0.0	-7.3	-7.4	24.3
Jm_09	1488.0	1543.6	1498.4	1494.3	1491.7	0.4	0.2	-3.2	-3.4	39.3
Jm_10	1714.1	1714.1	1748.9	1722.7	1714.1	0.5	0.0	0.5	0.0	36.6
					average	0.3	0.0	-1.3	-1.5	34.1

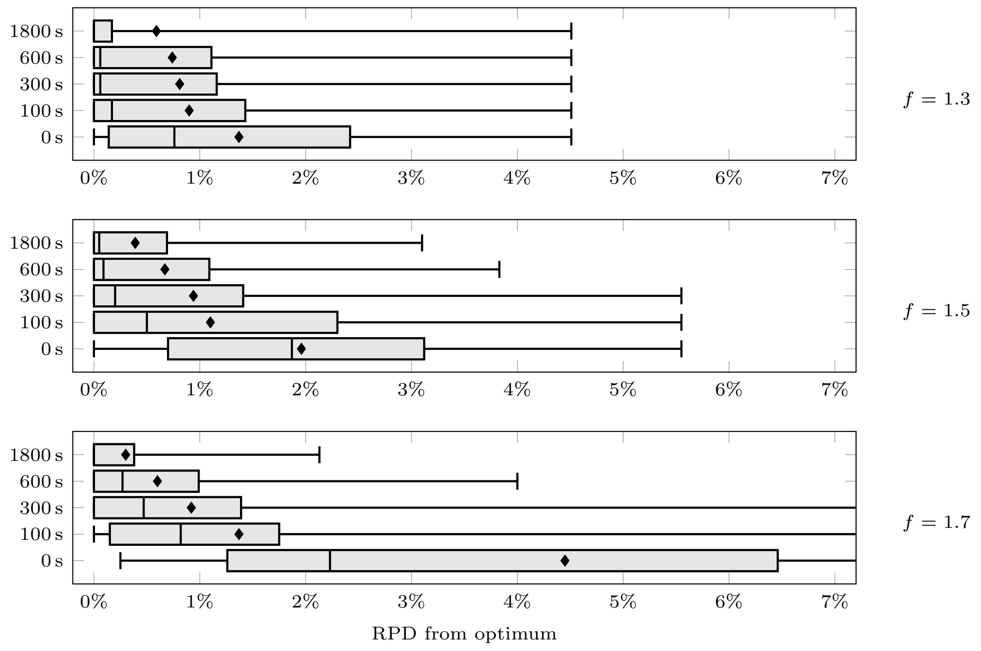


Figure 10: Box plots depicting the RPDs of the TS results with respect to the optimal objective values after 0, 100, 300, 600, and 1800 seconds for the small JIT-JS-LIN instances with $f = 1.3$ (top), $f = 1.5$ (middle), and $f = 1.7$ (bottom). The box plots are cropped at 7%.

solutions. In general, solutions of very high quality are already available within 300 s across all f-values; however, these plots clarify that the search afterwards is not in vain. The additional time helps TS attain a more robust set of solutions – as evident from the decrease of the median and the third quartile beyond 300 s.

5.2.2 Larger JIT-JS-LIN instances

We next analyze the results obtained for the larger JIT-JS-LIN instances with 100 operations. The detailed results appear in Tables 3-4. The MIP formulation is only able to solve 19 of these 66 instances to optimality and hits the time limit with a feasible solution otherwise. The optimality gaps are quite large in most of these cases. Therefore, differently from the previous section, in Figure 11 we choose to plot the progress of TS over time with respect to the BK values.

For the instances with $f = 1.3$, TS improves considerably over both MIP and BK. More specifically, the RPDs of the average objective value of the ten TS runs from the incumbent MIP solution and BK are -7.7% and -9.9% on average, respectively. To a large extent, this is due to the good quality of the initial solutions – see Figure 11, which in turn suggests that the due dates are quite tight. This can be confirmed by the last column of Table 3, which shows that the total tardiness cost comprises more than 50% of the overall cost in these instances. While the initial solutions are good, TS still further improves them by about 1.5% on average within the first minutes of the search – refer to Figure 11.

With looser due dates, the quality of the initial solutions degrade substantially as the earliness and intermediate storage holding costs start making up the larger share of the total cost. Based on Figure 11, we observe that the objective value of BK is on average about 8% lower than the average objective value of the initial solutions in TS for the instances with $f = 1.5$. However, TS quickly identifies substantially better solutions and ultimately terminates with solutions that are on average 2.9% better than those of BK. In terms of the best objective values attained by TS, the improvement over the current state-of-the-art heuristic algorithm BK is much more impressive with a margin of 7.7%. Furthermore, the average TS solutions perform on a par with those from MIP with a slight edge of 0.7% on average. The average RPD of the best TS solutions offer a further sizeable advantage of 4.7% to TS over MIP.

Table 3: Detailed results for the larger JIT-JS-LIN instances of Bülbül and Kaminsky (2013) with $f = 1.3$ (top) and $f = 1.5$ (bottom). The structure of the table is similar to that of Table 2, except that for MIP both a lower (lb) and an upper (ub) bound are reported at termination. Instances solved to optimality by MIP are indicated in bold in column 'ub' with no corresponding entry in column 'lb'.

$f = 1.3$

	MIP		BK	TS			RPD – ub		RPD – BK		tard.
inst	ub	lb	value	init	avg	best	avg	best	avg	best	%
abz05	19304.1	7792.9	15625.6	14088.0	13975.9	13319.1	-27.6	-31.0	-10.6	-14.8	50.6
abz06	**5806.2**		7344.2	5899.6	5852.3	5806.2	0.8	0.0	-20.3	-20.9	42.4
la16	5004.3	3033.6	5443.5	5704.3	5245.6	4947.2	4.8	-1.1	-3.6	-9.1	63.7
la17	**5402.0**		6562.6	6326.5	5884.5	5725.4	8.9	6.0	-10.3	-12.8	57.6
la18	5491.2	2617.6	5819.5	5118.0	5079.4	4982.2	-7.5	-9.3	-12.7	-14.4	60.4
la19	7995.5	4156.1	8436.9	6590.6	6542.7	6250.7	-18.2	-21.8	-22.5	-25.9	54.3
la20	3095.3	2869.3	3396.8	3738.3	3556.5	3422.8	14.9	10.6	4.7	0.8	48.9
la21	**3768.8**		4279.5	3802.3	3801.2	3800.7	0.9	0.8	-11.2	-11.2	34.1
la22	8155.8	7272.3	8995.9	8179.9	7991.8	7956.3	-2.0	-2.4	-11.2	-11.6	58.5
la23	3522.5	3451.0	4852.1	3597.3	3590.0	3527.8	1.9	0.2	-26.0	-27.3	36.7
la24	**4658.1**		5957.3	5045.9	4942.2	4678.4	6.1	0.4	-17.0	-21.5	54.5
mt10	13088.0	5072.1	11765.8	10650.4	10579.0	10379.2	-19.2	-20.7	-10.1	-11.8	73.1
orb01	12312	4828	10958.3	10245.3	10145.6	9684.3	-17.6	-21.3	-7.4	-11.6	64.9
orb02	7011.2	3539.5	5503.3	5568.4	5539.0	5477.0	-21.0	-21.9	0.6	-0.5	60.0
orb03	15027.9	4751.4	12992.3	12052.5	11988.3	11865.4	-20.2	-21.0	-7.7	-8.7	71.5
orb04	8148.7	3758.4	8313.5	8220.1	8117.7	7926.1	-0.4	-2.7	-2.4	-4.7	62.9
orb05	8080.1	3593.7	6601.4	6485.5	6443.2	5967.4	-20.3	-26.1	-2.4	-9.6	60.9
orb06	8157.7	3672.5	8733.9	7570.4	7523.3	7287.8	-7.8	-10.7	-13.9	-16.6	56.2
orb07	5866.8	2148.4	5107.1	4869.8	4849.1	4504.7	-17.3	-23.2	-5.1	-11.8	63.4
orb08	12815.2	7339.3	13133.8	12427.2	12270.3	10962.4	-4.3	-14.5	-6.6	-16.5	69.3
orb09	9602.6	5331.1	10809.4	9844.4	9803.4	9616.3	2.1	0.1	-9.3	-11.0	67.4
orb10	9922.6	4430.7	8509.4	7358.3	7340.5	7296.2	-26.0	-26.5	-13.7	-14.3	64.0
						average	-7.7	-10.7	-9.9	-13.0	58.0

$f = 1.5$

	MIP		BK	TS			RPD – ub		RPD – BK		tard.
inst	ub	lb	value	init	avg	best	avg	best	avg	best	%
abz05	11008.6	8565.6	11281.6	12530.8	11782.7	11116.4	7.0	1.0	4.4	-1.5	15.3
abz06	**5189.5**		5322.9	7479.6	5635.5	5434.6	8.6	4.7	5.9	2.1	22.3
la16	3526.4	3057.9	3936.4	4437.4	3764.7	3628.2	6.8	2.9	-4.4	-7.8	18.4
la17	**4812.1**		5292.2	6044.2	5192.1	4835.2	7.9	0.5	-1.9	-8.6	32.8
la18	3546.9	2764.4	3762.1	3821.8	3582.0	3511.5	1.0	-1.0	-4.8	-6.7	18.5
la19	4991.8	4307.6	5227.5	5813.1	5458.4	5053.4	9.3	1.2	4.4	-3.3	17.4
la20	2850.7	2611.1	3251.3	3614.3	3136.0	2963.6	10.0	4.0	-3.5	-8.8	17.3
la21	**3441.9**		3924.4	4536.5	3700.7	3597.9	7.5	4.5	-5.7	-8.3	5.5
la22	**5938.2**		7058.1	8363.4	6328.0	5938.2	6.6	0.0	-10.3	-15.9	26.6
la23	**3230.5**		3254.6	3733.6	3423.4	3334.6	6.0	3.2	5.2	2.5	8.3
la24	**3185.0**		3422.1	4132.8	3705.5	3379.5	16.3	6.1	8.3	-1.2	14.2
mt10	5908.8	4366.3	6222.1	6106.5	5961.0	5763.0	0.9	-2.5	-4.2	-7.4	33.4
orb01	9160	4208	9277.8	7282.6	7167.8	7027.5	-21.8	-23.3	-22.7	-24.3	36.1
orb02	4423.9	3779.1	4628.1	4802.9	4611.2	4510.1	4.2	1.9	-0.4	-2.5	10.1
orb03	12069.2	4094.8	8610.5	9159.6	9031.8	8119.5	-25.2	-32.7	4.9	-5.7	42.0
orb04	6201.9	3839.5	5844.5	6340.9	5705.5	5492.8	-8.0	-11.4	-2.4	-6.0	17.9
orb05	**4226.5**		4665.7	4698.2	4407.2	4269.1	4.3	1.0	-5.5	-8.5	12.4
orb06	7255.0	3860.0	6220.2	7622.5	6997.4	6353.4	-3.6	-12.4	12.5	2.1	38.3
orb07	3367.3	2342.7	3416.8	3823.5	3496.8	3351.4	3.8	-0.5	2.3	-1.9	12.4
orb08	13018.3	4858.6	10307.7	9079.0	8419.8	8134.8	-35.3	-37.5	-18.3	-21.1	54.3
orb09	8073.3	4106.9	7475.5	7059.6	6887.1	6416.9	-14.7	-20.5	-7.9	-14.2	37.5
orb10	5483.7	3493.4	6421.2	5243.2	5089.7	4984.8	-7.2	-9.1	-20.7	-22.4	27.6
						average	-0.7	-5.4	-2.9	-7.7	23.6

Table 4: Detailed results for the larger JIT-JS-LIN instances of Bülbül and Kaminsky (2013) with $f = 1.7$. See Tables 2 and 3 for a detailed description of the table structure.

$f = 1.7$

	MIP		BK	TS			RPD − ub		RPD − BK		tard.
inst	**ub**	**lb**	**value**	**init**	**avg**	**best**	**avg**	**best**	**avg**	**best**	**%**
abz05	11866.5	10207.0	11984.4	18378.4	11933.9	11875.9	0.6	0.1	-0.4	-0.9	9.7
abz06	**5752.2**		5939.6	10117.4	5946.9	5779.6	3.4	0.5	0.1	-2.7	2.2
la16	**3698.5**		3801.7	6105.3	4007.9	3701.3	8.4	0.1	5.4	-2.6	13.9
la17	**4031.2**		4117.6	6733.6	4475.9	4065.7	11.0	0.9	8.7	-1.3	1.1
la18	3523.4	3333.6	3803.7	5006.0	3623.1	3540.1	2.8	0.5	-4.7	-6.9	6.0
la19	5220.3	4939.2	5592.4	7757.4	5465.6	5327.0	4.7	2.0	-2.3	-4.7	0.8
la20	**3079.6**		3213.9	4496.0	3319.5	3095.3	7.8	0.5	3.3	-3.7	3.9
la21	**3708.6**		3975.0	6159.3	3871.8	3809.8	4.4	2.7	-2.6	-4.2	7.2
la22	5879.3	5280.8	6805.9	9756.1	6387.2	6042.1	8.6	2.8	-6.2	-11.2	6.3
la23	**3667.7**		3717.0	5577.8	3858.1	3692.5	5.2	0.7	3.8	-0.7	5.7
la24	**3670.9**		3778.6	6011.0	3762.4	3670.9	2.5	0.0	-0.4	-2.8	6.1
mt10	4786.9	4740.9	5181.0	6026.8	5169.8	4871.3	8.0	1.8	-0.2	-6.0	15.2
orb01	7253	4542	6749.6	8885.9	7873.4	7255.7	8.6	0.0	16.7	7.5	24.1
orb02	**4195.2**		4367.0	6594.4	4719.5	4219.8	12.5	0.6	8.1	-3.4	3.1
orb03	6384.4	4586.0	6913.0	8374.2	7191.4	7015.6	12.6	9.9	4.0	1.5	14.6
orb04	5219.7	4741.9	5494.4	6621.5	5701.4	5545.3	9.2	6.2	3.8	0.9	15.8
orb05	4419.0	3558.6	4539.4	6633.5	4522.0	4274.3	2.3	-3.3	-0.4	-5.8	13.4
orb06	6198.2	5016.5	6496.3	8387.6	6546.3	6445.1	5.6	4.0	0.8	-0.8	25.7
orb07	3136.2	2871.7	3157.0	4744.0	3310.5	3136.2	5.6	0.0	4.9	-0.7	9.9
orb08	7839.3	4689.9	8441.4	9932.8	7994.6	7508.4	2.0	-4.2	-5.3	-11.1	19.9
orb09	5677.6	5040.2	6037.9	7656.1	5889.4	5609.7	3.7	-1.2	-2.5	-7.1	12.2
orb10	4393.9	3877.6	4889.2	6452.7	5032.0	4828.0	14.5	9.9	2.9	-1.3	15.6
						average	6.5	1.6	1.7	-3.1	10.6

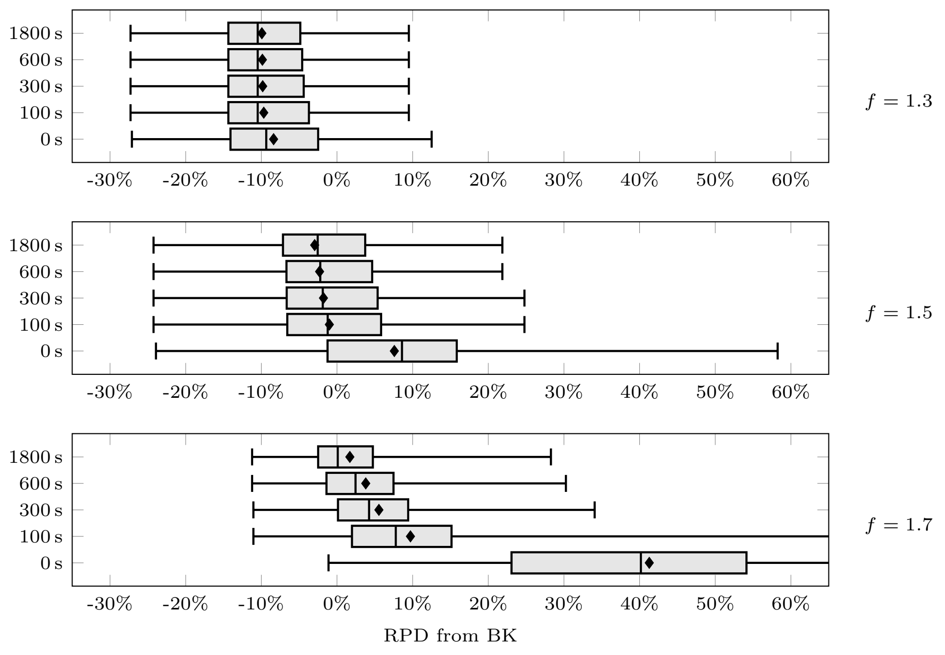


Figure 11: Box plots depicting the RPDs of the TS results with respect to the BK values after 0, 100, 300, 600, and 1800 seconds for the larger JIT-JS-LIN instances with $f = 1.3$ (top), $f = 1.5$ (middle), and $f = 1.7$ (bottom). The box plots are cropped at 60%.

The pattern crystallizes when we also take into account the results for $f = 1.7$. Both MIP and BK gain ground over TS with increasing f-values. In these instances, the tardiness costs contribute only about 10% to the overall costs, which unsurprisingly results in larger RPDs for the initial TS solutions – typically in the range from 25% to 55%. The good news is that marked improvements in solution quality accompany the first minutes of the search, and the quality of the final TS solutions is at least comparable to that of BK. The average TS objective values are somewhat inferior to the associated BK values with an average RPD of 1.7%; however, in 19 out of 22 instances at least one TS solution outperforms the corresponding BK solution. The average RPD for the best TS solution with respect to BK is -3.1%. Finally, observe that increasing f-values translate into easier instances as far as MIP is concerned. The number of optimal solutions obtained increases from 4 to 7 to 8 for $f = 1.3, 1.5, 1.7$, respectively. In light of this observation, it is fair to state that TS fares well against MIP even on instances with $f = 1.7$ – as justified by the average RPDs of 6.5% and 1.6% for the average and best TS solutions, respectively.

Based on the results so far on JIT-JS-LIN, a couple of concluding remarks about the performance of our approach is in order. First off, there is room for an alternate initial solution generation method for instances with loose due dates. Moreover, a more significant insight can be gleaned from studying Figures 10-11. On small instances, the run time in excess of 300 s is put into good use to reduce the variability of the TS solutions. Similarly, TS is still able to move the median by about 5% from 300 s to 1800 s on large instances with $f = 0.7$. In contrast, the progress pretty much stalls at 100 s for the other two f-values while solving the large instances. These observations imply that TS would certainly benefit from including a diversification scheme. However, in this paper we leave such more sophisticated mechanisms to be embedded into TS – or some other (meta)heuristic – as future research. The focus here is on the timing problem and the associated neighborhood definition, and we prefer to keep the structure of TS simple to be able to isolate the effect of these. Ultimately, even with a relatively basic TS implementation we get good results, and these can be attributed to the quality of the neighborhood definition. We confidently claim that TS is the new state-of-the-art heuristic for JIT-JS-LIN. It achieves better results than BK across the board both on small and large instances with some mixed results for large instances with $f = 1.7$. TS also compares favorably against MIP on large instances with $f = 1.3, 1.5$. On a final note, observe that the RPD of the best TS solution from the corresponding MIP solution on the 19 large instances MIP solves to optimality is just 1.75% on average with a maximum of 6.11%. This is definitely an accomplishment for a generic heuristic not tailored specifically to this difficult job shop scheduling problem with a non-regular objective.

5.3 JIT-JS-SUPERLIN

In this section, we delve into the results obtained for the JIT-JS-SUPERLIN instances in an effort to demonstrate the general applicability of our approach across both linear and non-linear objective functions. The detailed performance metrics are reported in Table 5, and Figure 12 depicts the improvement in solution quality over time for TS. The underlying benchmark data set from Lawrence (1984) in these experiments is quite diverse with the number of operations ranging from small instances with 50 operations to very large instances with 300 operations. As the original instances do not feature any due dates, these were produced as specified in Section 5.1. During this process, our intention has been to create a fair set of instances – recall a consistent pattern from Section 5.2.2 on larger instances of JIT-JS-LIN: loose due dates hurt the performance of our approach. In this context, both the f-value and the percentage of the total cost arising from the tardiness of the jobs in a best TS solution have been good indicators of instance difficulty for TS. If the value of the latter indicator goes above 50%, then TS is quite likely to identify near-optimal solutions, and lower values are associated with poorer RPDs. Ultimately, the f-values in this section were set diligently such that we cover a wide range of values generally below 50% for the share of the total tardiness cost.

In the absence of a viable competing algorithm in the literature capable of handling JIT-JS-SUPERLIN, TS is only benchmarked against the MIP formulation in this section. It turns out that JIT-JS-SUPERLIN is substantially more challenging for MIP compared to JIT-JS-LIN. Only two of the smallest 10×5 instances are solved to optimality, and the optimality gaps are large for most of the other instances. Nevertheless, MIP finds feasible solutions for all instances allowing us to compute the RPDs of the TS solutions with respect to the incumbents (upper bounds) from MIP.

Table 5: Detailed results for the JIT-JS-SUPERLIN instances. The structure of the table is identical to that of Table 3 – except that the f-values are specified on a per instance basis with an additional column, and all 'BK' columns related to Bülbül and Kaminsky (2013) are removed. The instances are grouped according to their size $n \times m$.

		MIP		TS			RPD − ub		tard.
inst	f	**ub**	**lb**	**init**	**avg**	**best**	**avg**	**best**	**%**
la01	1.9	**2137.6**		2963.8	2167.2	2165.4	1.4	1.3	46.5
la02	1.9	**1392.8**		1696.6	1413.7	1392.8	1.5	0.0	15.9
la03	1.9	1735.5	1513.3	1951.4	1832.6	1735.5	5.6	0.0	40.3
la04	1.9	2340.4	1359.4	2718.1	2381.6	2279.5	1.8	-2.6	65.2
la05	1.9	1980.5	1000.8	2611.4	2300.4	2076.8	16.2	4.9	49.4
10 × 5						**average**	5.3	0.7	43.4
la06	2.4	4678.1	1135.6	5680.8	4928.5	4684.4	5.4	0.1	54.2
la07	2.4	5661.5	996.4	6303.0	5235.3	5150.2	-7.5	-9.0	55.0
la08	2.4	3669.8	953.4	4264.3	3438.4	3344.5	-6.3	-8.9	28.8
la09	2.4	5995.0	1109.7	7695.7	5995.9	5784.6	0.0	-3.5	48.9
la10	2.4	4761.4	975.1	5679.5	4725.5	4527.4	-0.8	-4.9	47.4
15 × 5						**average**	-1.8	-5.2	46.9
la11	3.1	7588.2	1093.9	8605.9	7253.6	6441.5	-4.4	-15.1	26.5
la12	3.1	6993.6	1072.2	7848.7	6112.6	5724.7	-12.6	-18.1	34.1
la13	3.1	6476.0	1117.3	8080.7	6547.5	6003.3	1.1	-7.3	21.0
la14	3.1	10454.7	1288.6	12156.4	9700.6	9151.2	-7.2	-12.5	59.9
la15	3.1	10907.6	1206.2	11642.2	9788.3	8974.1	-10.3	-17.7	56.4
20 × 5						**average**	-6.7	-14.1	39.6
la16	1.4	2827.7	945.4	2768.8	2733.6	2716.3	-3.3	-3.9	37.5
la17	1.4	2870.5	1433.5	3190.5	3003.4	2893.5	4.6	0.8	54.8
la18	1.4	2340.3	914.3	2480.1	2340.6	2338.9	0.0	-0.1	36.3
la19	1.4	2719.7	870.6	2613.9	2599.5	2478.5	-4.4	-8.9	30.0
la20	1.4	2443.4	862.1	2425.3	2419.6	2390.5	-1.0	-2.2	30.6
10 × 10						**average**	-0.8	-2.8	37.9
la21	1.7	7234.6	995.9	6095.1	5852.1	5557.9	-19.1	-23.2	32.8
la22	1.7	7630.0	932.9	6834.5	6418.9	6128.7	-15.9	-19.7	41.0
la23	1.7	6557.5	1000.1	5977.8	5515.0	5287.1	-15.9	-19.4	22.3
la24	1.7	5534.2	930.3	5223.7	5027.0	4745.9	-9.2	-14.2	8.9
la25	1.7	5302.3	900.5	5600.1	5115.7	4411.8	-3.5	-16.8	20.7
15 × 10						**average**	-12.7	-18.7	25.1
la26	1.9	18909.7	1082.3	14033.1	13133.0	11337.6	-30.5	-40.0	39.3
la27	1.9	21662.8	1046.5	15808.2	15182.0	13427.6	-29.9	-38.0	42.5
la28	1.9	15551.7	1103.5	14606.3	14230.0	12327.9	-8.5	-20.7	49.8
la29	1.9	21125.5	1015.2	12680.6	11900.4	9345.2	-43.7	-55.8	41.6
la30	1.9	17047.0	1169.2	13626.1	13016.5	11093.1	-23.6	-34.9	46.3
20 × 10						**average**	-27.3	-37.9	43.9
la31	2.9	52747.3	1270.0	26816.5	23892.2	21885.6	-54.7	-58.5	24.7
la32	2.9	32498.5	1449.4	28694.6	26058.1	23451.9	-19.8	-27.8	12.9
la33	2.9	29592.4	1332.8	25238.5	22288.8	19500.2	-24.7	-34.1	23.0
la34	2.9	36616.7	1405.2	29894.4	27397.3	23899.7	-25.2	-34.7	18.2
la35	2.9	49623.4	1340.2	29491.2	26848.7	23651.6	-45.9	-52.3	24.9
30 × 10						**average**	-34.1	-41.5	20.7
la36	1.4	11873.0	968.1	11367.5	10756.6	9450.1	-9.4	-20.4	67.8
la37	1.4	15511.4	971.8	10131.7	9590.0	8616.2	-38.2	-44.5	51.1
la38	1.4	13466.6	959.2	8486.5	8148.5	7526.7	-39.5	-44.1	45.1
la39	1.4	7515.1	913.6	6486.0	6273.6	5144.6	-16.5	-31.5	26.5
la40	1.4	9677.7	893.2	7846.2	7587.7	6723.4	-21.6	-30.5	41.6
15 × 15						**average**	-25.0	-34.2	46.4

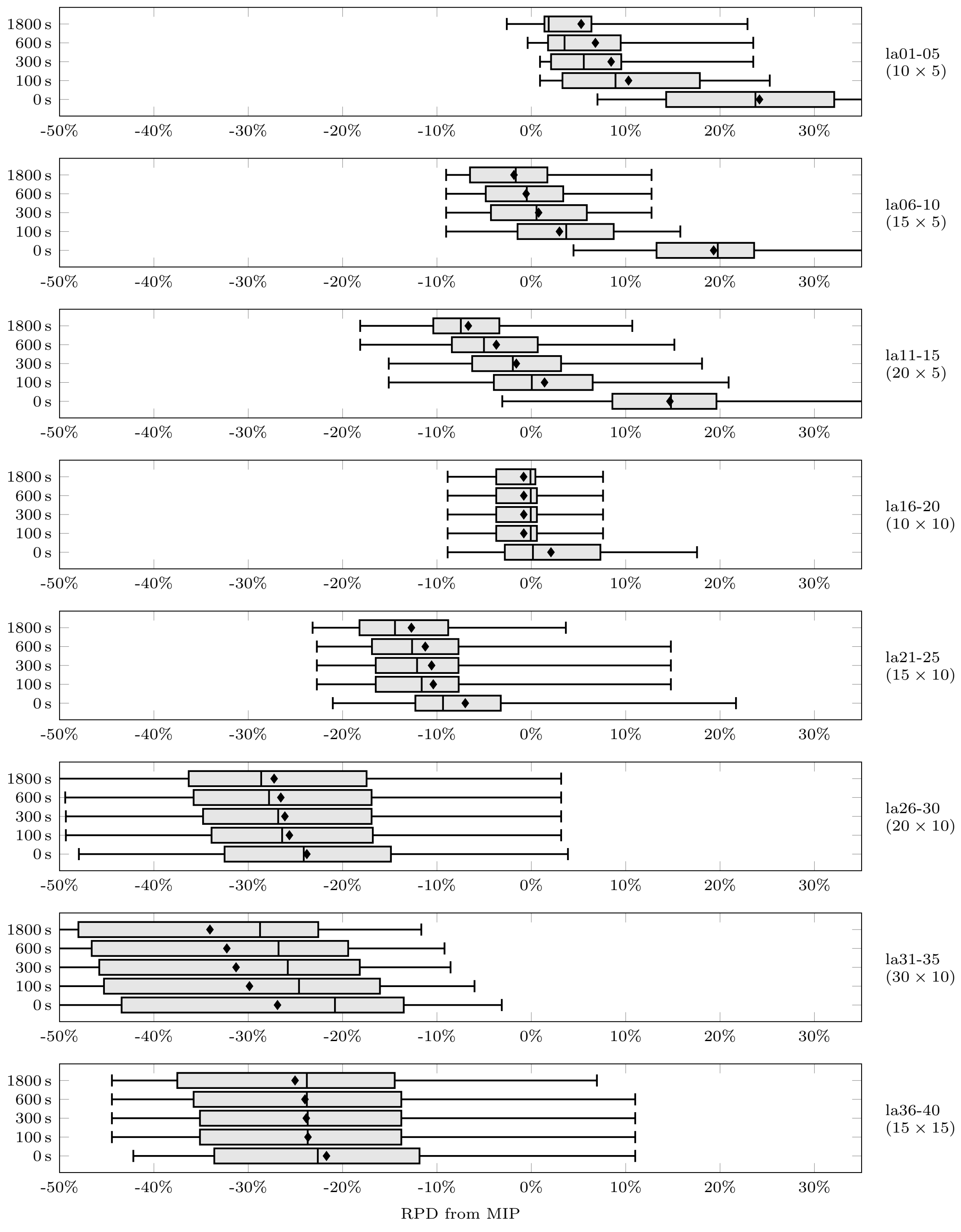


Figure 12: Box plots grouped by instance size depicting the RPDs of the TS results with respect to the incumbent solutions from MIP after 0, 100, 300, 600, and 1800 seconds for the JIT-JS-SUPERLIN instances. The box plots are cropped at -50% and 35%.

For the instances with five machines (i.e., la01 to la15), the quality of the initial solutions is quite low with a typical RPD of 10 to 30%. Interestingly, this is in stark contrast to the range 0.5% - 6.5% obtained for the small JIT-JS-LIN instances with 4 machines. Usual of TS, progress in solution quality is attained quickly in a couple of minutes. Compared to that of MIP, the solution quality of TS at termination goes from being somewhat worse for 10 jobs to slightly better for 15 jobs, and finally to considerably better for 20 jobs – even though the average share of the tardiness costs is lowest for the 20×5 instances. It appears that the disadvantage of our approach stemming from the relatively smaller share of the tardiness costs is more than offset by the growth in the instance sizes. This is a trend we did not get to observe in Section 5.2 because the sizes of the JIT-JS-LIN instances are constant.

A similar pattern is repeated for the instances with 10 and 15 machines (i.e., la16 to la40). TS and MIP perform similarly on instances of size 10×10, but from this point on TS outperforms MIP by increasingly larger margins as the number of jobs increases. The average RPD of the average TS solution per instance group is -0.8%, -12.7%, -27.3%, -34,1%, and -25.0% for sizes 10×10, 15×10, 20×10, 30×10, and 15×15, respectively. In a majority of these instances, even the initial TS solution is of higher quality than the terminal solution of MIP – which should probably be attributed to the inability of MIP to handle large instances. In any case, TS can back up the initial solution quality with further improvements during the search and exhibits a progressively dominant behavior against MIP as the instance size grows.

A couple of important observations stand out if we evaluate the results for JIT-JS-LIN and JIT-JS-SUPERLIN together. On the one hand, stalling is less of an issue in Figure 12 compared to the behavior in Figure 11. On the other hand, the variability in solution quality becomes significant for the larger instances in Figure 12. Both of these issues point to a need for a diversification scheme combined with a faster evaluation of the neighbors so that various broad areas in the feasible region can be scanned quickly in a quest for better solutions. Another insight to be underlined is that the instance size is the main determinant of how TS fares against competing methods. For a fixed instance size, the looseness of the due dates – also reflected by the contribution of the tardiness costs to the overall cost – plays to the disadvantage of TS. However, our approach is the clear choice for larger instances sizes with 150 operations or more and represents the state-of-the-art in job shop scheduling with non-regular linear and non-linear convex objectives, regardless of due date tightness.

6 Conclusion

JS–CONV is a generic job shop scheduling problem, where the objective is to minimize a sum of separable convex cost functions attached to the operation start times and the differences between the start times of arbitrary pairs of operations. This generic objective function makes it possible to address not only the typical objectives pursued in the literature, such as the makespan or the total weighted tardiness, but also more complex non-linear and non-regular objectives, including, for example, convex earliness, storage, and tardiness cost terms. Furthermore, the objective function allows for the integration of some less conventional managerial considerations, such as the prioritization of job completion times via modeling constructs beyond objective weights, and certain process characteristics, such as the perishability of intermediate products.

Based on a comprehensive combinatorial problem formulation on a disjunctive graph, we developed a tabu search heuristic for **JS–CONV** where neighbors are built by swapping a critical arc. The criticality of the arcs is assessed by optimally solving a timing problem, which structurally corresponds to the dual of a network flow problem with convex costs. As a solution method, we apply the cost-scaling algorithm proposed in (Ahuja et al., 2003). We show that the timing problem is amenable to quick solution times, so that it can successfully be integrated into local search approaches. We executed extensive numerical tests on just-in-time job shop scheduling instances with linear and non-linear costs and obtained results supporting the validity of the proposed approach.

We believe that these developments pave the way for interesting future research. From a methodological viewpoint, the tabu search may benefit from an additional diversification mechanism as discussed in Section 5. One could simply start some tabu search runs with different initial solutions in parallel, so as to efficiently exploit today's multi-core processors, or include more elaborate diversification strategies, such as path relinking, which was successfully applied in various machine scheduling problems (see, e.g., Nowicki and Smutnicki, 2005; Vallada and Ruiz, 2010; Peng et al., 2015). As discussed in Section 5.1, most of the computation time of the tabu search is spent in the timing problems, which are solved from scratch each time. Per similar discussions in (Avci, 2001; Avci and Storer, 2004), one may develop a "warm-start" strategy for the applied cost-scaling network flow algorithm. More specifically, as we are attacking a series of similar timing problems in the tabu search, one may study how to efficiently leverage the past calculations for solving the timing problems in subsequent iterations. In our framework, we may draw upon the ideas of Vidal et al. (2015). Nevertheless, we expect that solving the timing problem for **JS–CONV** will always remain a computational bottleneck in local search methods, substantially limiting the number of iterations. Hence, it is important to start any local search for **JS–CONV** with a good initial solution and keep only the promising neighbors in the neighborhood. Therefore, one may develop constructive heuristics for **JS–CONV** based on the dispatching and insertion heuristics of Wennink (1995), or devise a compact neighborhood following the move quality analysis of Avci and Storer (2004).

From an application perspective, one could address various interesting job shop scheduling problems in the **JS–CONV** framework. In addition to the applications mentioned so far, one may consider, for example, controllable processing times. While these times are considered to be constant in most scheduling problems, they are occasionally controllable –at least to some extent– by adjusting the allocation of resources to processing, such as energy, manpower, money, or subcontracting (Shabtay and Steiner, 2007). Various practical applications of this process feature are discussed in the literature: Janiak (1989), for example, considers a scheduling problem in steel mills, Trick (1994) refers to scheduling with tooling machines, and Aktürk et al. (2010) address the scheduling of computer numeric control (CNC) turning operations. Scheduling with controllable processing times has received considerable attention from researchers. However, the literature on solution methods for job shop type problems with controllable processing times is scarce according to a recent literature survey of Shabtay and Steiner (2007). We suggest to study the application of **JS–CONV** to controllable processing times with a so-called locally bounded resource consumption. This can easily be modeled in **JS–CONV** by adding a node for the end time of each operation to the disjunctive graph and defining appropriate convex processing cost functions for the time difference between the start and end of each operation. Finally, one could integrate more complex process features into **JS–CONV**, such as no-storage constraints (so-called blocking), transportation operations, or flexibility in the choice of the machines. With these features, the timing problem may still be solvable by the same approach as in **JS–CONV**. However, swapping individual critical arcs only may lead to infeasible neighbors. Job-insertion-based neighborhoods, which were already applied successfully in various complex job shop scheduling problems (Gröflin et al., 2011; Bürgy and Gröflin, 2016, 2017) could be employed to preserve feasibility of the neighborhood.

Appendix

We first provide the proofs of Proposition 1 and Lemma 1 and then describe the MIP formulation applied for our computational tests.

Proof. (of Proposition 1) Among all optimal timing solutions, let $\boldsymbol{\alpha}$ be an earliest start time solution, i.e., there exists no optimal solution $\boldsymbol{\alpha}' \neq \boldsymbol{\alpha}$ such that $\alpha'_i \leq \alpha_i$ for all $i \in \mathcal{I}$.

If $\alpha_i \leq U$ holds for all $i \in \mathcal{I}$, we are done. Otherwise, build the following graph. Add a fictive start node σ and a node i for each operation $i \in \mathcal{I}$. Add an arc (i, j) if we cannot decrease the start time of j by one time unit without possibly increasing the costs associated to the start time difference between i and j or rendering the solution infeasible. Specifically, for each operation $i \in I$, add an arc (σ, i) with length $\lceil \alpha_i^{\min} \rceil$ if $\alpha_i \leq \lceil \alpha_i^{\min} \rceil$. Indeed, we could decrease the start time of i by one time unit without increasing the costs $b_i(\alpha_i)$ if $\alpha_i > \lceil \alpha_i^{\min} \rceil$ since b_i is convex and has its minimum at $\alpha_i^{\min}$. Similarly, for each $(i, j) \in S \cup (A \setminus Q)$, add an arc (i, j) with length p_i if $\alpha_j - \alpha_i = p_i$, for each $(i, j) \in Q \setminus A$, add an arc (i, j) with length $\lceil \delta_{ij}^{\min} \rceil$ if $\alpha_j - \alpha_i \leq \lceil \delta_{ij}^{\min} \rceil$, and for each $(i, j) \in Q \cup A$, add an arc (i, j) with length $\max(p_i, \lceil \delta_{ij}^{\min} \rceil)$ if $\alpha_j - \alpha_i \leq \max(p_i, \lceil \delta_{ij}^{\min} \rceil)$.

Consider the constructed graph. If some nodes are not reachable from σ by a path, then a new optimal solution $\boldsymbol{\alpha}'$ is obtained by reducing the start time α_i for all non-reachable nodes i by one time unit. But then $\boldsymbol{\alpha}' \neq \boldsymbol{\alpha}$ and $\alpha_i' \leq \alpha_i$ for all $i \in \mathcal{I}$, which contradicts the choice of $\boldsymbol{\alpha}$.

In the other case, all nodes are reachable from σ by a path, and the length of each path from σ to a node i provides an upper bound for the start time α_i as the time difference $\alpha_j - \alpha_i$ is at most the length of arc (i,j) for all arcs introduced in the graph and $\alpha_\sigma = 0$. However, by construction, these paths cannot be longer than $U = \max_{i \in P}(\lceil \alpha_i^{\min} \rceil) + \sum_{i \in \mathcal{I}} p_i + \sum_{(i,j) \in Q} \lceil \delta_{ij}^{\min} \rceil$. Hence, $\alpha_i \leq U$ must hold for all $i \in \mathcal{I}$. □

Proof. (of Lemma 1) Suppose there exists an arc $e = (i,j) \in S \setminus S^{\text{crit}}$ with $x_e \neq 0$. As arc e is not critical, $x_e > 0$ is implied. Then, there exist a backward arc (j,i) in the residual network $Res(\boldsymbol{x}, S)$. We will show that there exists a negative-length cycle containing (j,i) in $Res(\boldsymbol{x}, S)$, which, using the optimality conditions (6), contradicts that x is optimal.

Consider the timing graph $T(S)$. By the flow balance constraint (5b) at node j and as $x_{ij} > 0$, there exists an arc (j, w_1) with $x_{j,w_1} > 0$ or an arc (w_1, j) with $x_{w_1,j} < 0$. In both cases, the flow balance constraint (5b) at node w_1 implies that there exists an arc (w_1, w_2) with $x_{w_1,w_2} > 0$ or an arc (w_2, w_1) with $x_{w_2,w_1} < 0$. Repeating this procedure, we finally must arrive at node i. This means that we constructed a walk from j to i using arcs with positive flow in forward direction and arcs with negative flow in backward direction. Hence, there exists an (undirected) path with node set $(j = v_0, v_1, ..., v_r = i)$ using arcs in this way. As all arcs with positive (negative) flow give rise to a backward (forward) arc in the residual network $Res(\boldsymbol{x}, S)$, $\mathcal{C} = (j, i = v_r, v_{r-1}, ..., v_0 = j)$ is (the node set of) a cycle in $Res(\boldsymbol{x}, S)$.

We show that the length of cycle $\mathcal{C}$ is negative. First, arc (j,i) has length $-U$ as the left slope at x_{ij} is U (see (5d) and Figure 6). Second, look at all backward arcs (w,v) in $\mathcal{C}$. The associated arc (v,w) in timing graph $T(S)$ has flow $x_{vw} > 0$. By (5d), function $C_{vw}(x_{vw}) = -F_{vw}(q) + q x_{vw}$ for some q with $0 < x_{vw} \leq a_{vw}(q) = F_{vw}(q+1) - F_{vw}(q)$. Hence q must be so that the value of function F_{vw} increases from q to $q+1$. Therefore q must be at least $\lfloor \alpha_w^{\min} \rfloor$ if $v = \sigma$, p_v if $(v,w) \in A$ and $\lfloor \delta_{vw}^{\min} \rfloor$ if $(v,w) \in Q \setminus A$. The length of arc (w,v) in $\mathcal{C}$, which is the negative of the left slope of C_{vw} at x_{vw}, is then at most $-q$, hence it is at most $-\lfloor \alpha_w^{\min} \rfloor$ if $v = \sigma$, $-p_v$ if $(v,w) \in A$ and $-\lfloor \delta_{vw}^{\min} \rfloor$ if $(v,w) \in Q \setminus A$. As $\alpha_w^{\min} \geq 0$, $p_v > 0$ and $\delta_{vw}^{\min} \geq 0$, all costs associated to backward arcs are non-positive. Third, similarly, we can show that for each forward arcs (v,w) in $\mathcal{C}$, its length is at most $\lceil \alpha_w^{\min} \rceil$ if $v = \sigma$, p_v if $(v,w) \in A \setminus Q$, $\lceil \delta_{vw}^{\min} \rceil$ if $(v,w) \in Q \setminus A$ and $\max(p_i, \lceil \delta_{vw}^{\min} \rceil)$ if $(v,w) \in Q \cap A$.

In summary, the length of arc (j,i) is $-U = -\max_{v \in P}(\lceil \alpha_v^{\min} \rceil) - \sum_{v \in \mathcal{I}} p_v - \sum_{(v,w) \in Q} \lceil \delta_{vw}^{\min} \rceil$ by (3). If $\mathcal{C}$ visits node σ, the total length of the two arcs incident to σ is not higher than $\max_{v \in P}(\lceil \alpha_v^{\min} \rceil)$, and for each other arc (v,w) in $\mathcal{C}$, its length is not larger than $\max(p_v, \lceil \delta_{vw}^{\min} \rceil)$. Then, it is easy to see that cycle C must be of negative length. Finally, observe that $\sum_{(v,w) \text{ in } \mathcal{C}} c_{vw}^{\pi}$ is exactly the length of cycle $\mathcal{C}$, But by (6), this must be non-negative. Hence, if $x_e > 0$, x cannot be an optimal flow. □

MIP formulation The following MIP was applied for the computational tests. We introduce a binary variable x_{ij} for all distinct operations $i, j \in \mathcal{I}$ with $M(i) = M(j)$ and $J(i) \neq J(j)$. If $x_{ij} = 1$, operation i is executed before j; otherwise, $x_{ij} = 0$ and operation j is executed before i. To capture the earliness and tardiness values, we introduce two non-negative continuous variables β_K^{early} and β_K^{tardy} for each job K. Finally, set Q contains all pairs (i,j) of consecutive operations in all jobs. Then, we obtain the following MIP formulation for JIT-JS-LIN:

$$\text{minimize} \sum_{(i,j) \in Q} c_{ij}^{\text{storInt}}(\delta_{ij} - p_i) + \sum_{K \in \mathcal{J}} \left(c_K^{\text{stor}}(\alpha_{K_1} - r_K) + c_K^{\text{early}} \beta_K^{\text{early}} + c_K^{\text{tardy}} \beta_K^{\text{tardy}} \right) \tag{7a}$$

subject to:

$$\alpha_{K_1} \geq r_K \text{ for all } K \in \mathcal{J}, \tag{7b}$$

$$\alpha_j - \alpha_i \geq p_i \text{ for all } (i,j) \in Q, \tag{7c}$$

$$\begin{aligned} &\alpha_j - \alpha_i \geq p_i - U(1 - x_{ij}) \text{ and} \\ &\alpha_i - \alpha_j \geq p_j - U x_{ij} \text{ for all } i,j \in \mathcal{I} \text{ with } M(i) = M(j), J(i) \neq J(j), \end{aligned} \tag{7d}$$

$$\alpha_j - \alpha_i - \delta_{ij} = 0 \text{ for all } (i,j) \in Q, \tag{7e}$$

$$\beta_K^{\text{early}} + \alpha_i \geq d_K - p_i \text{ and}$$

$$\alpha_i - \beta_K^{\text{tardy}} \leq d_K - p_i \text{ for all } K \in \mathcal{J} \text{ and } i = K_{|K|}, \tag{7f}$$

$$\beta_K^{\text{early}}, \beta_K^{\text{tardy}} \geq 0 \text{ for all } K \in \mathcal{J}, \tag{7g}$$

$$\alpha_i \in \mathbb{Z}^* \text{ for all } i \in \mathcal{I}, \tag{7h}$$

$$x_{ij} \in \{0,1\} \text{ for all } i,j \in \mathcal{I} \text{ with } M(i) = M(j), J(i) \neq J(j). \tag{7i}$$

The objective (7a) minimizes the sum of the weighted earliness/tardiness and the intermediate storage holding costs. Clearly, the first term in the objective (7a) can be replaced with $\sum_{(i,j)\in Q} c_{ij}^{\text{storInt}} \delta_{ij}$ by omitting the constant $\sum_{(i,j)\in Q} -c_{ij}^{\text{storInt}} p_i$. The correspondence of the constraints (7c)-(7e) to (1b)-(1d) is obvious – except that the disjunctive machine capacity constraints (1c) are linearized above in constraints (7d) through the use of the sequencing variables $\mathbf{x}$ and the large constant U denoting the latest start time for any operation as defined in Section 3. The extra constraints (7b) mandate that no first operation in any job starts before the associated job's release time. Furthermore, (7f) ensure that $\alpha_{K_{|K|}} + p_{K_{|K|}} + \beta_K^{\text{early}} - \beta_K^{\text{tardy}} = d_K$ for all $K \in \mathcal{J}$ given the structure of these constraints and the positive penalty coefficents associated with the earliness and tardiness variables in the objective (7a).

For JIT-JS-SUPERLIN, the superlinear tardiness costs $(\beta_K^{\text{tardy}})^l$, $K \in \mathcal{J}$ are linearized as follows. For each job $K \in \mathcal{J}$, we introduce a continuous variable z_K representing the tardiness cost for job K. In the objective function (7a), the term $c_K^{\text{tardy}} \beta_K^{\text{tardy}}$ is then replaced by z_K. The value of z_K is bounded from below by adding to (7) a linear constraint

$$z_K \geq a + b\beta_K^{\text{tardy}} \text{ with } a = t^l - bt, b = (t+1)^l - t^l$$

for each time point $t \in \{0, ..., U + \max_{K\in\mathcal{J}} p_{K_{|K|}}\}$. It can easily be verified that $\max_{t\in\{0,...,U+\max_{K\in\mathcal{J}} p_{K_{|K|}}\}} a + b\beta_K^{\text{tardy}} = (\beta_K^{\text{tardy}})^l$ in an optimal solution.

Note that quadratic tardiness costs corresponding to $l = 2$ can directly be specified in the objective function for `Gurobi` to handle. For other values of $l > 1$, a direct specification is not possible and a linearization technique, such as the one above, is typically applied.

References

J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401, 1988.

R. H. Ahmadi and U. Bagchi. Minimizing job idleness in deadline constrained environments. *Operations Research*, 40(5):972–985, 1992.

R. K. Ahuja, D. S. Hochbaum, and J. B. Orlin. Solving the convex cost integer dual network flow problem. *Management Science*, 49(7):950–964, 2003.

M. S. Aktürk, A. Atamtürk, and S. Gürel. Parallel machine match-up scheduling with manufacturing cost considerations. *Journal of Scheduling*, 13(1):95–110, 2010.

P. Amorim, H. Meyr, C. Almeder, and B. Almada-Lobo. Managing perishability in production-distribution planning: A discussion and review. *Flexible Services and Manufacturing Journal*, 25(3):389–413, 2013.

S. Avci. *Algorithms for Generalized Nonregular Scheduling Problems*. PhD thesis, Lehigh University, 2001.

S. Avci and R. H. Storer. Compact local search neighborhoods for generalized scheduling problems. Technical Report 04T-001, Industrial and Systems Engineering, Lehigh University, 2004.

E. Balas. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, 17(6):941–957, 1969.

E. Balas and A. Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management Science*, 44(2):262–275, 1998.

P. Baptiste, M. Flamini, and F. Sourd. Lagrangian bounds for just-in-time job-shop scheduling. *Computers & Operations Research*, 35(3):906–915, 2008.

J. C. Beck and P. Refalo. Combining local search and linear programming to solve earliness/tardiness scheduling problems. In N. Jussien and F. Laburthe, editors, *Proceedings of the Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, pages 221–235, 2002.

J. C. Beck and P. Refalo. A hybrid approach to scheduling with earliness and tardiness costs. *Annals of Operations Research*, 118:49–71, 2003.

C. Bierwirth and J. Kuhpfahl. Extended GRASP for the job shop scheduling problem with total weighted tardiness objective. *European Journal of Operations Research*, 261(3):835–848, 2017.

P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41:157–183, 1993.

P. Brandimarte and M. Maiocco. Job shop scheduling with a non-regular objective: A comparison of neighbourhood structures based on a sequencing/timing decomposition. *International Journal of Production Research*, 37(8):1697–1715, 1999.

K. Bülbül. A hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem. *Computers & Operations Research*, 38(6):967–983, 2011.

K. Bülbül and P. Kaminsky. A linear programming-based method for job shop scheduling. *Journal of Scheduling*, 16(2):161–183, 2013.

K. Bülbül, P. Kaminsky, and C. Yano. Flow shop scheduling with earliness, tardiness and intermediate inventory holding costs. *Naval Research Logistics*, 51(3):407–445, 2004.

R. Bürgy. A neighborhood for complex job shop scheduling problems with regular objectives. *Journal of Scheduling*, 20(4):391–422, 2017.

R. Bürgy and H. Gröflin. The blocking job shop with rail-bound transportation. *Journal of Combinatorial Optimization*, 31(1):152–181, 2016.

R. Bürgy and H. Gröflin. The no-wait job shop with regular objective: A method based on optimal job insertion. *Journal of Combinatorial Optimization*, 33(3):977–1010, 2017.

H. Chen and P. B. Luh. An alternative framework to lagrangian relaxation approach for job shop scheduling. *European Journal of Operational Research*, 149(3):499–512, 2003.

B.-C. Cheng, A. D. Stoyenko, T. J. Marlowe, and S. K. Baruah. LSTF: A new scheduling policy for complex real-time tasks in multiple processor systems. *Automatica*, 33(5):921–926, 1997.

T. Cheng and J. Jiang. Job shop scheduling for missed due-date performance. *Computers & Industrial Engineering*, 34(2):297–307, 1998.

K. De Bontridder. Minimizing total weighted tardiness in a generalized job shop. *Journal of Scheduling*, 8 (6):479–496, 2005.

M. Drótos, G. Erdős, and T. Kis. Computing lower and upper bounds for a large-scale industrial job shop scheduling problem. *European Journal of Operational Research*, 197(1):296–306, 2009.

S. Eilon and R. Hodgson. Job shops scheduling with due dates. *International Journal of Production Research*, 6(1):1–13, 1967.

I. Essafi, Y. Mati, and S. Dauzère-Pérès. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. *Computers & Operations Research*, 35(8):2599–2616, 2008.

K. Fagerholt. A computer-based decision support system for vessel fleet scheduling - experience and future research. *Decision Support Systems*, 37(1):35–47, 2004.

P. Farahani, M. Grunow, and H.-O. Günther. Integrated production and distribution planning for perishable food products. *Flexible Services and Manufacturing Journal*, 24(1):28–51, 2012.

S. Gélinas and F. Soumis. Dantzig-wolfe decomposition for job shop scheduling. In *Column Generation*, pages 271–302. Springer, 2005.

F. Glover. Tabu search and adaptive memory programming - advances, applications and challenges. *Interfaces in Computer Science and Operations Research*, 7:1–75, 1996.

F. W. Glover and M. Laguna. *Tabu Search*. Kluwer, 1997.

T. C. Gonçalves, J. M. S. Valente, and J. E. Schaller. Metaheuristics for the single machine weighted quadratic tardiness scheduling problem. *Computers & Operations Research*, 70:115–126, 2016.

M. Á. González, I. González-Rodríguez, C. R. Vela, and R. Varela. An efficient hybrid evolutionary algorithm for scheduling with setup times and weighted tardiness minimization. *Soft Computing*, 16(12):2097–2113, 2012.

R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

D. Grimes and E. Hebrard. Solving variants of the job shop scheduling problem through conflict-directed search. *INFORMS Journal on Computing*, 27(2):268–284, 2015.

H. Gröflin, D. N. Pham, and R. Bürgy. The flexible blocking job shop with transfer and set-up times. *Journal of Combinatorial Optimization*, 22(2):121–144, 2011.

D. J. Hoitomt, P. B. Luh, E. Max, and K. R. Pattipati. Scheduling jobs with simple precedence constraints on parallel machines. *IEEE Control Systems Magazine*, 10(2):34–40, 1990.

J. Hurink and S. Knust. Tabu search algorithms for job-shop problems with a single transport robot. *European Journal of Operational Research*, 162(1):99–111, 2005.

A. Jain and S. Meeran. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113(2):390–434, 1999.

A. Janiak. Minimization of the blooming mill standstills - mathematical model, suboptimal algorithms. *Mechanika*, 8(2):37–49, 1989.

P. Jaskowski and A. Sobotka. Using soft precedence relations for reduction of the construction project duration. *Technological and Economic Development of Economy*, 18(2):262–279, 2012.

C. A. Kaskavelis and M. C. Caramanis. Efficient lagrangian relaxation algorithms for industry size job-shop scheduling problems. *IIE Transactions*, 30(11):1085–1097, 1998.

T. Kis. Job-shop scheduling with processing alternatives. *European Journal of Operational Research*, 151(2): 307–332, 2003.

S. Kreipl. A large step random walk for minimizing total weighted tardiness in a job shop. *Journal of Scheduling*, 3(3):125–138, 2000.

J. Kuhpfahl and C. Bierwirth. A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research*, 66:44–57, 2016.

S. Lawrence. Supplement to resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA, 1984.

Y. Mati, S. Dauzère-Pérès, and C. Lahlou. A general approach for optimizing regular criteria in the job-shop scheduling problem. *European Journal of Operational Research*, 212(1):33–42, 2011.

H. Matsuo, C. Suh, and R. Sullivan. A controlled simulated annealing method for the general job shop scheduling problem. Technical Report 03-04-88, Department of Management, Graduate School of Business, The University of Texas at Austin, Austin, TX, 1988.

J.-N. Monette, Y. Deville, and P. Van Hentenryck. Just-in-time scheduling with constraint programming. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 241–248, 2009.

K. G. Murty. *Linear and Combinatorial Programming.* John Wiley & Sons, 1976.

K. Neumann, C. Schwindt, and N. Trautmann. Advanced production scheduling for batch plants in process industries. *OR Spectrum*, 24(1):251–279, 2002.

E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

E. Nowicki and C. Smutnicki. An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, 8(2):145–159, 2005.

H. Ohta and T. Nakatanieng. A heuristic job-shop scheduling algorithm to minimize the total holding cost of completed and in-process products subject to no tardy jobs. *International Journal of Production Economics*, 101(1):19–29, 2006.

N. R. Parsa, B. Karimi, and S. M. Husseini. Exact and heuristic algorithms for the just-in-time scheduling problem in a batch processing system. *Computers & Operations Research*, 80:173–183, 2017.

B. Peng, Z. Lü, and T. C. E. Cheng. A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research*, 53:154–164, 2015.

F. Peng and Y. Ouyang. Track maintenance production team scheduling in railroad networks. *Transportation Research Part B: Methodological*, 46(10):1474–1488, 2012.

M. Pinedo and M. Singer. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Naval Research Logistics*, 46(1):1–17, 1999.

G. P. Prastacos. Blood inventory management: An overview of theory and practice. *Management Science*, 30(7):777–800, 1984.

B. Roy and B. Sussmann. Les problèmes d'ordonnancement avec contraintes disjonctives. Note de synthese et formation. No. 9, SEMA, Paris, France, 1964.

N. M. Sadeh. Focused simulated annealing search: An application to job shop scheduling. *Annals of Operations Research*, 63(1):77–103, 1996.

D. Shabtay and G. Steiner. A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666, 2007.

M. Singer and M. Pinedo. A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. *IIE Transactions*, 30(2):109–118, 1998.

M. A. Trick. Scheduling multiple variable-speed machines. *Operations Research*, 42(2):234–248, 1994.

R. J. M. Vaessens, E. H. Aarts, and J. K. Lenstra. Job shop scheduling by local search. *INFORMS Journal on Computing*, 8(3):302–317, 1996.

E. Vallada and R. Ruiz. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, 38(1-2):57–67, 2010.

W.-J. Van Hoeve, C. Gomes, B. Selman, and M. Lombardi. Optimal multi-agent scheduling with constraint programming. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI)*, pages 1813–1818, 2007.

P. J. M. Van Laarhoven, E. H. L. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):113–125, 1992.

L. A. Vidal, E. Sahin, N. Martelli, M. Berhoune, and B. Bonan. Applying AHP to select drugs to be produced by anticipation in a chemotherapy compounding unit. *Expert Systems with Applications*, 37(2):1528–1534, 2010.

T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. Timing problems and algorithms: Time decisions for sequences of activities. *Networks*, 65(2):102–128, 2015.

M. Wennink. *Algorithmic Support for Automated Planning Boards*. PhD thesis, Eindhoven University of Technology, 1995.

G. I. Zobolas, C. D. Tarantilis, and G. Ioannou. Exact, heuristic and meta-heuristic algorithms for solving shop scheduling problems. In F. Xhafa and A. Abraham, editors, *Metaheuristics for Scheduling in Industrial and Manufacturing Applications*, chapter 1, pages 1–40. Springer Berlin Heidelberg, 2008.