UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

DEPARTEMENT D'INFORMATIQUE
DEPARTEMENT FÜR INFORMATIK
Bd de Pérolles 90
CH-1700 Fribourg

www.unifr.ch/informatics

WORKING PAPER

The no-wait job shop with regular objective: a method based on optimal job insertion

R. Bürgy & H. Gröflin

Internal working paper no 15-02

July 2015

The no-wait job shop with regular objective: a method based on optimal job insertion

Reinhard Bürgy Heinz Gröflin

July 10, 2015

Abstract

The no-wait job shop problem (NWJS-R) considered here is a version of the job shop scheduling problem where, for any two operations of a job, a fixed time lag between their starting times is prescribed. Also, sequence-dependent set-up times between consecutive operations on a machine can be present. The problem consists in finding a schedule that minimizes a general regular objective function.

We study the so-called optimal job insertion problem in the NWJS-R and prove that this problem is solvable in polynomial time by a very efficient algorithm, generalizing a result we obtained in the case of a makespan objective.

We then propose a large neighborhood local search method for the NWJS-R based on the optimal job insertion algorithm and present extensive numerical results that compare favorably with current benchmarks when available.

1 Introduction

The landscape of job shop scheduling is characterized by a broad variety of process features—set-up times, blocking or no-wait constraints, transportation operations, to name a few—but also by a multitude of objectives pursued in finding "good" schedules.

An important objective is to minimize makespan and a large part of the job shop scheduling literature addresses this goal. On the other hand, from an operations management perspective on scheduling, other objectives are of equal interest, e.g. in the presence of release and due dates, schedules might be sought that minimize average flow time or some measure of lateness or tardiness.

The class of so-called regular objectives comprises all functions that are monotone non-increasing in the completion times. This class includes all the objectives mentioned previously, as well as many others. It is therefore of high interest, from both a theoretical and an application point of view, to develop solution methods for job shop scheduling problems with general regular objective.

Work along this line is very sparse in the literature, as noted by Mati et al. in [5]. They propose a general approach for the classical job shop scheduling

problem with regular objective. For more complex job shop scheduling problems, e.g. with blocking or no-wait constraints, solution methods for general regular objective do not seem to be available in the literature.

In the present work, we propose a method for the no-wait job shop scheduling problem with general regular objective (NWJS-R). The method is of the local search type with a large neighborhood and is based on the study of the optimal job insertion problem in the NWJS-R (OJI-NWJS-R). We establish that the OJI-NWJS-R is solvable in polynomial time by an efficient algorithm and apply repeatedly this algorithm to determine optimal neighbors in our local search.

It should be noted that we used a similar approach in previous work on the NWJS with makespan objective [2]. However, there are some subtle differences between the OJI-NWJS with makespan objective and the OJI-NWJS-R. Some structural properties holding in the first case and that were critical in the development of the algorithm in [2] are lost in the second, more general case. Fortunately and at first unexpectedly for us, the OJI-NWJS-R is still solvable in polynomial time, and in fact with a similar computational effort. Moreover, the proposed algorithm works for any regular function and requires only function evaluation calls.

The paper is organized as follows. The next section describes the NWJS-R and formulates it in both a classical disjunctive graph and a compact disjunctive graph. Section 3 is devoted to the OJI-NWJS-R. After its formulation in an insertion graph, its so-called feasible insertions are characterized as the stable sets (of prescribed cardinality) of a conflict graph H, and structural properties of H and the family of feasible insertions of bounded objective value are established. An efficient algorithm for the OJI-NWJS-R is then developed. Section 4 proposes a local search method for the NWJS-R based on the OJI-NWJS-R algorithm and presents extensive numerical results. The Appendix provides a detailed implementation of the OJI-NWJS-R algorithm and a complexity analysis.

In the exposition of the paper, we have tried to keep the paper self-contained: for this reason, several basic results from [2] are recalled and at times detailed for readability and insight.

We conclude this introduction with some notation and terminology. All graphs will be directed and the following standard notation will be used. In the graph $G = (V, E)$, an arc $e \in E$ has a *tail* (node) $t(e)$ and a *head* $h(e)$. For any disjoint sets M, $N \subseteq V$, $\delta(M, N) = \{e \in E : t(e) \in M \text{ and } h(e) \in N\}$, and for any $N \subseteq V$, $\delta(N) = \delta(N, V - N) \cup \delta(V - N, N)$. If an arc length vector $c \in R^E$ is given, G will be denoted by the triplet $G = (V, E, c)$. Sometimes a triplet alone is used to identify a graph, usually a subgraph of a given graph. In $G = (V, E, c)$, a cycle is called *positive* if its length is positive. Finally, some concepts such as clique, stable set, and comparability graph, are used here with directed graphs, with the understanding that they apply to the corresponding undirected graphs obtained by ignoring arc orientation.

2 The no-wait job shop

2.1 Problem description

Typically in a job shop scheduling problem, a set I of operations and a set M of machines are given. Each operation $i \in I$ needs a specific machine, say $m_i \in M$, for its execution (without interruption) of duration $p_i > 0$. The set of operations is structured into jobs: a set $\mathcal{J} \subseteq 2^I$ of jobs such that $\mathcal{J}$ forms a partition of I is given. For each job $J \in \mathcal{J}$, its set of operations $\{i : i \in J\}$ is usually ordered in a sequence $\{J_1, J_2, \ldots, J_{|J|}\}$, J_r denoting the r-th operation of job J. Two operations i, j of job J are consecutive if $i = J_r$ and $j = J_{r+1}$ for some $r, 1 \leq r < |J|$. The problem consists in finding starting times for all operations $i \in I$ so that each machine is occupied by at most one operation at a time and some objective function, e.g. the makespan, is minimized.

The no-wait job shop problem with regular objective (NWJS-R) considered here is the following version of a job shop scheduling problem. *No-wait constraints* are present in the following form. For each job $J \in \mathcal{J}$ and any two operations i and $j \in J$, a *fixed time lag* γ_{ij} of arbitrary sign is imposed between the starting times of i and j. We may assume that the order of operations $J_1, J_2, \ldots, J_{|J|}$ of job J is such that $\gamma_{J_r, J_{r+1}} \geq 0$, $1 \leq r < |J|$ and that time lags are given only between consecutive operations, since for any $i = J_s$ and $j = J_t$ with $s < t$, $\gamma_{ij} = \sum_{s \leq r < t} \gamma_{J_r, J_{r+1}}$ and $\gamma_{ji} = -\gamma_{ij}$. Note that fixed time lags slightly generalize the no-wait constraints present in the "classical" no-wait job shop where $\gamma_{ij} = p_i$ for a pair i, j of consecutive operations of a job. They allow to model some scheduling problems typically occurring in the process industry.

Two additional features, set-up times and objective function, are best described by first adding to I two dummy operations σ and τ, both of duration zero, where σ must precede all operations and τ be preceded by all operations, extending hereby I to $I^+ = I \cup \{\sigma, \tau\}$.

If i and j are two operations on a same machine and j follows i, then a *set-up* of duration s_{ij} might occur between completion of i and start of j. Also, for each $i \in I$, an *initial set-up* of duration $s_{\sigma i}$ between start/completion of σ and start of i, i.e. a *release time*, and similarly, a *final set-up* of duration $s_{i\tau}$ between completion of i and start of τ, i.e. a so-called *tail*, might be prescribed.

Let $\boldsymbol{\alpha} = (\alpha_i \in \mathbb{R} : i \in I^+)$ denote the vector of the starting times α_i, $i \in I^+$. The objective function considered here is a function f that satisfies

$$\boldsymbol{\alpha} \leq \boldsymbol{\alpha'} \Rightarrow f(\boldsymbol{\alpha}) \leq f(\boldsymbol{\alpha'}).$$

Such an objective function is called *regular*. (Note that in the literature, e.g. in [7], a regular function is often defined in terms of completion times β_i instead of starting times α_i, $i \in I^+$. Of course, since $\beta_i = \alpha_i + p_i$, both definitions are valid.)

The NWJS-R consists in finding starting times α_i for all operations $i \in I^+$ so that each machine is occupied by at most one operation at a time and the regular function f is minimized.

An example with four jobs J, K, L, N and five machines $m_1, \ldots, m_5$ is illustrated in a Gantt chart in Figure 1, upper part. For simplicity, no set-ups are

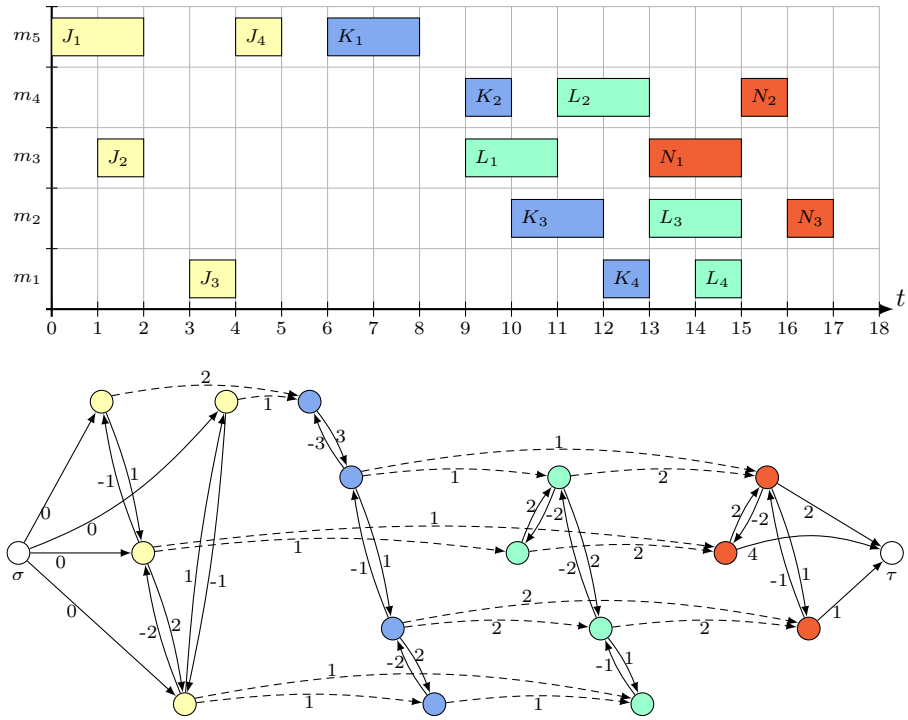

Figure 1: An example with four jobs and five machines.

present. The numerical data can be read in the chart, e.g. for job J, its first operation is executed on m_5 and has a duration of 2, and the time lag between its first and second operation is 1. We will use this example in the sequel.

2.2 A disjunctive graph formulation

As in the classical job shop problem, a disjunctive graph $G = (I^+, A, E, \mathcal{E}, d)$ for the NWJS-R is readily obtained as follows. Each operation $i \in I^+ = I \cup \{\sigma, \tau\}$ is represented by a node. Identifying a node with the operation it represents, we denote the node set by $I^+ = I \cup \{\sigma, \tau\}$.

The set A of conjunctive arcs consists of the following arcs: (i) for each $i \in I$, an initial set-up arc (σ, i) and a final set-up arc (i, τ) of respective length $d_{\sigma i} = s_{\sigma i}$ and $d_{i\tau} = p_i + s_{i\tau}$; (ii) for each job J and each ordered pair of consecutive operations $i, j \in J$, a pair of arcs (i, j) and (j, i) with respective length $d_{ij} = \gamma_{ij}$ and $d_{ji} = \gamma_{ji} = -d_{ij}$.

The set E of disjunctive arcs consists of all arcs (i, j) and (j, i) between operations i and j on a same machine and of different jobs. Formally, define for all $m \in M$, $I_m = \{i \in I : m_i = m\}$ and $E_m = \{(i, j) : i, j \in I_m$ such that $i \in J$, $j \in J' \Rightarrow J \neq J'\}$. Then $E = \cup_{m \in M} E_m$. The lengths are $d_{ij} = p_i + s_{ij}$ for all $(i, j) \in E$.

For any $m \in M$ and $i, j \in I_m$, arcs (i, j) and (j, i) form a (unordered) *pair* of disjunctive arcs. The family $\mathcal{E}$ is the collection of all such pairs. A general element of $\mathcal{E}$, i.e. a pair of disjunctive arcs, will be denoted by $\{e, \bar{e}\}$.

Definition 1. *Any subset of disjunctive arcs $S \subseteq E$ is called a selection. A selection S is positive acyclic if the subgraph $(I^+, A \cup S, d)$ contains no positive cycle, and is positive cyclic otherwise. A selection S is complete if $S \cap \{e, \overline{e}\} \neq \emptyset$ for all $\{e, \overline{e}\} \in \mathcal{E}$. A selection S is feasible if it is positive acyclic and complete.*

In Figure 1, lower part, the feasible selection (set of dashed arcs) corresponding to the schedule of the example above is depicted.

Given a feasible selection $S \subseteq E$, the space of feasible starting times is

$$\Omega(S) = \{\boldsymbol{\alpha} \in \mathbb{R}^{I^+} : \alpha_\sigma = 0;\ \alpha_{h(e)} - \alpha_{t(e)} \geq d_e \text{ for all } e \in A \cup S\}.$$

Since $(I^+, A \cup S, d)$ contains no positive cycle, $\Omega(S) \neq \emptyset$ and *earliest starting times* $\alpha(S) = (\alpha_i(S) : i \in I^+)$ can be calculated by determining for each $i \in I^+$ the length of a longest path from σ to i in $(I^+, A \cup S, d)$. Moreover, since the objective function f is regular,

$$f(\alpha(S)) = \min\{f(\boldsymbol{\alpha}) : \boldsymbol{\alpha} \in \Omega(S)\}.$$

The NWJS-R with regular objective f can therefore be formulated as the following problem in the disjunctive graph G: "Among all feasible selections, find a selection S minimizing $f(\alpha(S))$."

A remark on set-up times is in order. They should satisfy the so-called weak triangle inequality (cf. [1], p. 11) for the disjunctive graph formulation to be valid. Otherwise, arcs between non-consecutive operations on a machine may become active when computing longest paths in $(V, A \cup S, d)$, yielding wrong starting times since set-ups take place only between consecutive operations on a machine.

In a no-wait job shop, the starting time of any operation of a job determines the starting times of all other operations of that job, so that starting times defined on the jobs are sufficient. Accordingly, a more compact disjunctive graph formulation is readily obtained, as used by Schuster in [9]. For our purpose however, a different compact disjunctive graph formulation derived in [2] will be needed.

2.3 A compact disjunctive graph formulation

Given any two distinct jobs $J, K \in \mathcal{J}$ and any selection $S \subseteq E$ in the disjunctive graph $G = (I^+, A, E, \mathcal{E}, d)$, define

$$\begin{gathered} S_{JK} = S \cap \delta(J, K),\ S_{KJ} = S \cap \delta(K, J) \\ S_{[JK]} = S_{JK} \cup S_{KJ} \end{gathered} \tag{1}$$

and distances

$$c^S_{JK} = \max\{\gamma_{J_1,i} + d_{ij} + \gamma_{j,K_1} : (i,j) \in S_{JK}\}, \tag{2}$$

convening $c^S_{JK} = -\infty$ if $S_{JK} = \emptyset$ and $\gamma_{ii} = 0$ for all $i \in I$.

Observe that $\bigcup_{J,K \in \mathcal{J}} S_{[JK]}$ is a partition of S. Also, a distance c^S_{JK} $(> -\infty)$ is the length of a longest path in the subgraph $(I^+, A \cup S_{JK}, d)$ from the first

operation J_1 of J to the first operation K_1 of K, and similarly c^S_{KJ} ($> -\infty$) is the length of a longest path in $(I^+, A \cup S_{KJ}, d)$ from K_1 to J_1.

The distances from σ to J and from J to τ for all $J \in \mathcal{J}$ are defined as:

$$c_{\sigma J} = c^S_{\sigma J} = \max\{d_{\sigma i} + \gamma_{i,J_1} : i \in J\}, \tag{3}$$

$$c_{J\tau} = c^S_{J\tau} = \max\{\gamma_{J_1,i} + d_{i\tau} : i \in J\}. \tag{4}$$

A compact disjunctive graph formulation of the NWJS-R is now derived as follows. For each (unordered) pair of distinct jobs $J, K \in \mathcal{J}$, let $S^p_{[JK]} \subseteq \delta(J,K) \cup \delta(K,J)$, $p = 1, \ldots, q_{JK}$, be all selections that are positive acyclic and complete on $\delta(J,K) \cup \delta(K,J)$, i.e. $S^p_{[JK]} \cap \{e, \overline{e}\} \neq \emptyset$ for all $\{e, \overline{e}\} \subseteq \delta(J,K) \cup \delta(K,J)$. In other words, selections $S^p_{[JK]}$, $p = 1, \ldots, q_{JK}$, represent all feasible ways of positioning J and K with respect to each other.

The number q_{JK} of these selections is not larger than $1 + \sum_{m \in M} r_{J_m} \cdot r_{K_m}$, if r_{J_m} denotes the number of operations of J on machine m. In particular, in the case of a classical NWJS where $r_{J_m} \leq 1$ for all $J \in \mathcal{J}$ and $m \in M$, $q_{JK} \leq |M| + 1$.

We may assume that the $S^p_{[JK]}$'s are indexed with $p = 1, \ldots, q_{JK}$ in such a way that

$$\begin{aligned} &S^1_{JK} = S^1_{[JK]},\ S^1_{KJ} = \emptyset, \\ &S^p_{JK} \supset S^q_{JK} \text{ and } S^p_{KJ} \subset S^q_{KJ} \text{ for } 1 \leq p < q \leq q_{JK}, \\ &S^{q_{JK}}_{JK} = \emptyset,\ S^{q_{JK}}_{KJ} = S^{q_{JK}}_{[JK]}. \end{aligned} \tag{5}$$

i.e. $S^1_{[JK]}$ places job J "fully before" K, and, with increasing p, K moves ahead of an operation of J on some machine, until with selection $S^{q_{JK}}_{[JK]}$, K is fully before J.

Figure 2 illustrates these selections for the two jobs J and K in the example. The four positionings of J and K are depicted in a Gantt chart (left), and the corresponding selections $S^p_{[J,K]}, p = 1, 2, 3, 4$, are shown in the center (set of dashed arcs).

Proposition 2. *Let c^p_{JK} and c^p_{KJ} be the lengths c^S_{JK} and c^S_{KJ} defined by (2) for $S = S^p_{[JK]}$, $p = 1, \ldots, q_{JK}$. The following holds.*
i) $c^1_{JK} > c^2_{JK} > \ldots > c^{q_{JK}}_{JK}$ and $c^1_{KJ} < c^2_{KJ} < \ldots < c^{q_{JK}}_{KJ}$,
ii) $c^p_{JK} + c^p_{KJ} \leq 0$ for $p = 1, \ldots, q_{JK}$,
iii) $c^p_{JK} + c^q_{KJ} \leq 0$ for $1 \leq q < p \leq q_{JK}$,
iv) $c^p_{JK} + c^q_{KJ} > 0$ for $1 \leq p < q \leq q_{JK}$.

Proof. See [2]. □

The *compact disjunctive graph* $F = (\mathcal{J}^+, B, U, \mathcal{P}, c)$ is now constructed. The node set $\mathcal{J}^+ = \mathcal{J} \cup \{\sigma, \tau\}$ consists of all nodes representing a job or a fictive operation. The conjunctive arc set B comprises the arcs (σ, J) and (J, τ) with lengths $c_{\sigma J}$ and $c_{J\tau}$ defined by (3) and (4) for all $J \in \mathcal{J}$. The set U of disjunctive arcs comprises the following arcs. Between any distinct nodes J, K of $\mathcal{J}$, two arcs $(J,K)_p$ and $(K,J)_p$ with length c^p_{JK} and c^p_{KJ} are introduced for each $p = 1, \ldots, q_{JK}$.

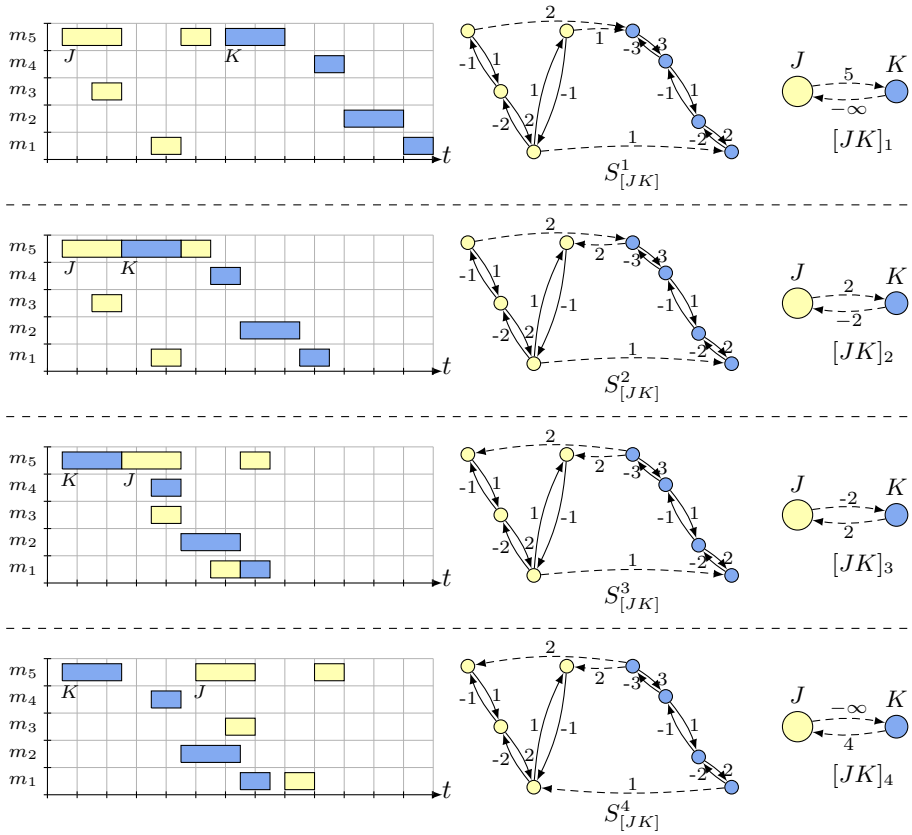

Figure 2: Positionings of jobs J and K (*left*) and the corresponding selections (set of *dashed arcs*) in graph G (*center*) and graph F (*right*).

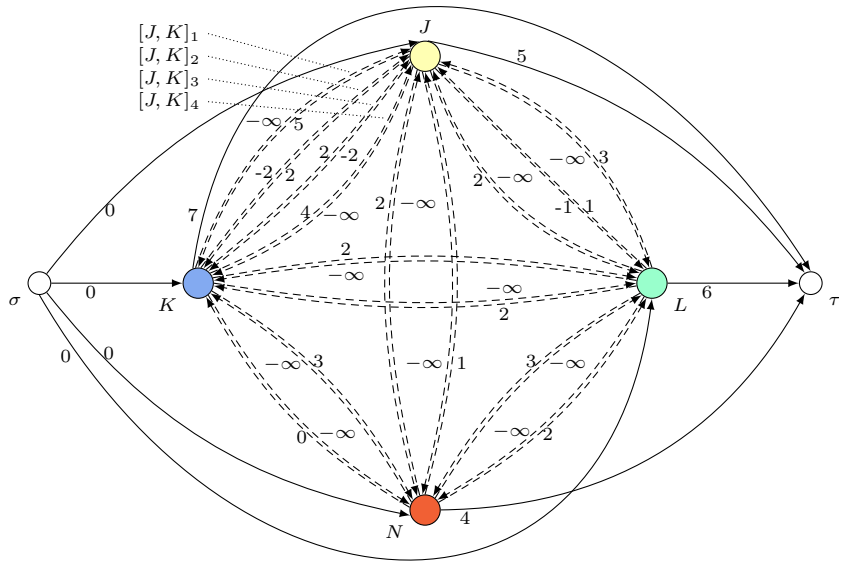

Figure 3: The disjunctive graph F of the example.

Considering jobs J and K in the example, two arcs $(J,K)_p$ and $(K,J)_p$ are introduced for $p = 1,2,3,4$. Each pair of arcs is separately displayed in Figure 2, right. The disjunctive graph F is illustrated in Figure 3.

For any distinct $J, K \in \mathcal{J}$ and $p \in \{1,\ldots,q_{JK}\}$, denote by $[J,K]_p$ the (unordered) *pair* of arcs $\{(J,K)_p, (K,J)_p\} \in U \times U$. The pair $[J,K]_p$ represents selection $S^p_{[JK]}$. Let P be the set of all such pairs, i.e.

$$P = \{[J,K]_p : J,K \in \mathcal{J},\ J \neq K \text{ and } 1 \leq p \leq q_{JK}\},$$

and for any distinct J and $K \in \mathcal{J}$, let

$$D_{JK} = \{[J,K]_p : 1 \leq p \leq q_{JK}\}.$$

The family $\mathcal{P}$ (of sets of arc pairs) is the family $\{D_{JK} : J,K \in \mathcal{J},\ J \neq K\}$.

Definition 3. *A selection in F is any set $T \subseteq P$ of arc pairs and $U_T \subseteq U$ denotes the set of all arcs used by T. A selection T is complete if $T \cap D_{JK} \neq \emptyset$ for all distinct $J,K \in \mathcal{J}$, and is positive acyclic if the graph $(\mathcal{J}^+, B \cup U_T, c)$ contains no positive cycle. T is feasible if it is complete and positive acyclic.*

Note that, by definition, a complete selection T contains at least one pair $[J,K]_p$ for any distinct jobs J and K. If additionally T is positive acyclic, i.e. if T is feasible, then by Proposition 2, T contains *exactly one* pair $[J,K]_p$ for distinct J and K.

Given a feasible selection $T \subseteq P$, the space of feasible starting times is

$$\Omega(T) = \{\boldsymbol{\alpha} \in \mathbb{R}^{\mathcal{J}^+} : \alpha_\sigma = 0;\ \alpha_{h(u)} - \alpha_{t(u)} \geq c_u \text{ for all } u \in B \cup U_T\}.$$

Since $(\mathcal{J}^+, B \cup U_T, c)$ contains no positive cycle, earliest starting times $\alpha(T) = (\alpha_J(T) : J \in \mathcal{J}^+)$ can be calculated by determining for each $J \in \mathcal{J}^+$ the length of a longest path from σ to J. Moreover, since f is regular,

$$f(\alpha(T)) = \min\{f(\boldsymbol{\alpha}) : \boldsymbol{\alpha} \in \Omega(T)\}.$$

The NWJS-R with regular objective f can then be formulated as the following problem in the compact disjunctive graph F: "Among all feasible selections, find a selection T minimizing $f(\alpha(T))$."

3 Optimal job insertion

3.1 The disjunctive insertion graph

Inserting optimally a given job can be thought of as the problem: given a feasible selection for all other jobs, insert the job in such a way that the resulting schedule is feasible and its objective value is minimal.

As in [2], we formulate the optimal job insertion problem in the NWJS with regular objective (OJI-NWJS-R) in the framework of our compact formulation. The job to be inserted will be denoted J, and for brevity, $\mathcal{J}^-$ and q_K will designate $\mathcal{J} - J$ and q_{JK}.

In the disjunctive graph F, a selection R for all other jobs $K \in \mathcal{J}^-$ is given which is positive acyclic and "complete", i.e. for any distinct $K, L \in \mathcal{J}^-$, $[K, L]_p \subseteq R$ for some $p \in \{1, \ldots, q_{KL}\}$. Let $U^J = U \cap \delta(J)$ and $\mathcal{P}^J$ be the family of sets D_{JK} for all $K \in \mathcal{J}^-$. One can define the *insertion graph* for J as the disjunctive graph $F^J = (\mathcal{J}^+, B \cup U_R, U^J, \mathcal{P}^J, c)$, where the restriction of c to $B \cup U_R \cup U^J$ is denoted again by c. A selection T in F^J is called an *insertion* of job J. Note that T is a (positive acyclic, complete, feasible) insertion if and only if $T \cup R$ is a (positive acyclic, complete, feasible) selection in $F = (\mathcal{J}^+, B, U, \mathcal{P}, c)$.

The OJI-NWJS-R can then be stated as follows: "Among all feasible insertions, find an insertion T minimizing $f(\alpha(T))$, where $\alpha(T) = (\alpha_K(T) : K \in \mathcal{J}^+)$ and $\alpha_K(T)$ is the length of a longest path from σ to K, $K \in \mathcal{J}^+$, in the subgraph $(\mathcal{J}^+, B \cup U_R \cup U_T, c)$."

3.2 Insertions and stable sets in the conflict graph

In the graph $(\mathcal{J}^+, B \cup U_R, c)$, which contains no positive cycle, let l_{KL} be the length of a longest K-L–path for any nodes K, $L \in \mathcal{J}^+$, with the convention $l_{KL} = 0$ if $K = L$ and $l_{KL} = -\infty$ if $K \neq L$ and there is no K-L–path. Obviously,

$$l_{KL} + l_{LK} \leq 0 \text{ and } l_{KL} + l_{LN} \leq l_{KN} \text{ for all } K, L, N \in \mathcal{J}^+. \tag{6}$$

Also, $l_{\sigma J} = c_{\sigma J}$ and $l_{J\tau} = c_{J\tau}$ where $c_{\sigma J}$ and $c_{J\tau}$ are defined by (3) and (4).

Definition 4. *The conflict graph is the graph $H = (W, Y)$ with node set W and arc set Y defined by $W = \cup_{K \in \mathcal{J}^-} W_K$ where $W_K = \{w_K^p : p = 1, \ldots, q_K\}$, $K \in \mathcal{J}^-$, and for any w_K^p, $w_L^q \in W$:*

$$(w_K^p, w_L^q) \in Y \Leftrightarrow c_{JK}^p + c_{LJ}^q + l_{KL} > 0 \tag{7}$$

Observe that if $K = L$, (7) is equivalent to:

$$(w_K^p, w_K^q) \in Y \Leftrightarrow p < q. \tag{8}$$

Indeed, by Proposition 2, $c_{JK}^p + c_{KJ}^q > 0$ if and only if $p < q$. Also, $l_{KK} = 0$ so that in (7) $c_{JK}^p + c_{LJ}^q + l_{KL} > 0$ for $K = L$ is equivalent to $p < q$. Therefore each W_K, $K \in \mathcal{J}^-$, is (the node set of) a *clique* in H.

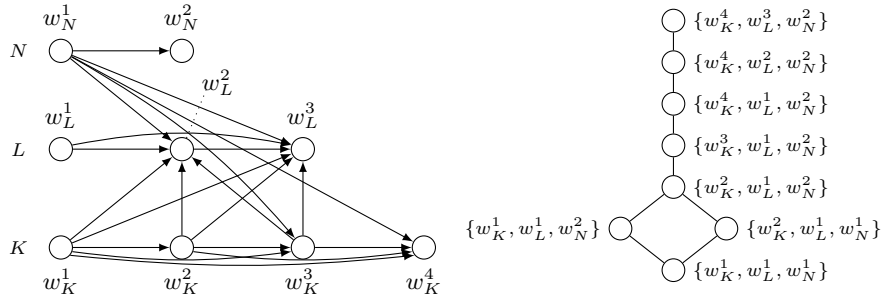

Figure 4: In the example, (left) the conflict graph H and (right) the Hasse diagram of lattice $\mathcal{L}$.

Figure 4, left, illustrates the conflict graph H of the example.

Conflict graphs allow to characterize feasible insertions as the following result holds.

Theorem 5. *There is a one-to-one correspondence between the feasible insertions and the stable sets of size $|\mathcal{J}^-|$ in H.*

Proof. Let T be a feasible insertion. Since T is complete and positive acyclic, given any $K \in \mathcal{J}^-$, $[J,K]_p \subseteq T$ for exactly one $p \in \{1,\ldots,q_K\}$, say p_K. Hence to T corresponds the node set $T' = \{w_K^{p_K} : K \in \mathcal{J}^-\} \subseteq W$ of size $|\mathcal{J}^-|$ in H. T' is stable in H. Indeed, suppose the contrary: for some $K \neq L$, $(w_K^{p_K}, w_L^{p_L}) \in Y$, i.e. $c_{JK}^{p_K} + c_{LJ}^{p_L} + l_{KL} > 0$. Then the positive cyclic insertion $\{[J,K]_{p_K}, [L,J]_{p_L}\}$ is contained in T, contradicting T being positive acyclic.

Conversely, let T' be a stable set of size $|\mathcal{J}^-|$ in H. T' picks up exactly one node, say $w_K^{p_K}$, from each clique W_K, $K \in \mathcal{J}^-$, hence $T' = \{w_K^{p_K} : K \in \mathcal{J}^-\}$. The corresponding insertion $T = \cup_{K\in\mathcal{J}^-}[J,K]_{p_K}$ is obviously complete. T is also positive acyclic, otherwise there is a positive cycle in $(\mathcal{J}^+, B \cup U_R \cup U_T, c)$ which must go through J, entering J, through, say, arc $(L,J)_{p_L}$ and leaving J through $(J,K)_{p_K}$, implying $c_{JK}^{p_K} + c_{LJ}^{p_L} + l_{KL} > 0$, hence $(w_K^{p_K}, w_L^{p_L}) \in Y$, a contradiction to the stability of T'. □

The conflict graph H has nice properties. First, H is a *comparability graph*.

Theorem 6. *$H = (W,Y)$ is acyclic and transitively oriented.*

Proof. a) H is transitively oriented, i.e. for any distinct nodes w, w', $w'' \in W$,

$$(w,w') \in Y \text{ and } (w',w'') \in Y \Rightarrow (w,w'') \in Y.$$

Indeed, assume $(w,w') = (w_K^p, w_L^q) \in Y$ and $(w',w'') = (w_L^q, w_N^r) \in Y$. Then $c_{JK}^p + c_{LJ}^q + l_{KL} > 0$ and $c_{JL}^q + c_{NJ}^r + l_{LN} > 0$. Adding both left and right hand sides of the two inequalities and using $c_{JL}^q + c_{LJ}^q \leq 0$ and $l_{KL} + l_{LN} \leq l_{KN}$ yields $c_{JK}^p + c_{NJ}^r + l_{KN} > 0$, hence $(w_K^p, w_N^r) \in Y$.

b) H is acyclic. Since H is transitively oriented, it suffices to show that there is no $w, w' \in W$ with both (w,w') and $(w',w) \in Y$. Assume the contrary and let $w = w_K^p$ and $w' = w_L^q$. Then $c_{JK}^p + c_{LJ}^q + l_{KL} > 0$ and $c_{JL}^q + c_{KJ}^p + l_{LK} > 0$. Since $c_{JK}^p + c_{KJ}^p \leq 0$ and $c_{JL}^q + c_{LJ}^q \leq 0$, $l_{KL} + l_{LK} > 0$ follows, a contradiction to (6). □

Second, the stable sets of size $|\mathcal{J}^-|$ in H have the following property.

Proposition 7. *The stable sets of size $|\mathcal{J}^-|$ in H form a lattice $\mathcal{L}$ with order $\prec$, meet $\vee$ and join $\wedge$ defined as follows. For any two stable sets $T = \{w_K^{p_K} : K \in \mathcal{J}^-\}$ and $T' = \{w_K^{p'_K} : K \in \mathcal{J}^-\}$,*

$$T \preceq T' \;\Leftrightarrow\; p_K \leq p'_K \text{ for all } K \in \mathcal{J}^-$$
$$T \vee T' = \{w_K^{\max\{p_K, p'_K\}} : K \in \mathcal{J}^-\}$$
$$T \wedge T' = \{w_K^{\min\{p_K, p'_K\}} : K \in \mathcal{J}^-\}$$

Proof. Clearly, $\prec$ is a partial order. $T \vee T'$ is stable, otherwise there exist w and $w' \in T \vee T'$ with $(w, w') \in Y$ or $(w', w) \in Y$. We may assume $w \in T - T'$ and $w' \in T' - T$, i.e. $w = w_K^{p_K}$ for some K and $p_K > p'_K$ and $w' = w_L^{p'_L}$ for some $L \neq K$ and $p'_L > p_L$. If $(w, w') = (w_K^{p_K}, w_L^{p'_L}) \in Y$, then $(w_K^{p'_K}, w_L^{p'_L}) \in Y$ by $(w_K^{p'_K}, w_K^{p_K}) \in Y$, and transitivity, and if $(w', w) = (w_L^{p'_L}, w_K^{p_K}) \in Y$, then $(w_L^{p_L}, w_K^{p_K}) \in Y$, contradicting the stability of T' or T. Similarly, one can show that $T \wedge T'$ is stable. Finally, $|T \vee T'| = |T \wedge T'| = |T| = |T'|$. □

We remark that this result also follows from the fact that the family of maximum size stable sets in a comparability graph forms a lattice (see for instance [8], p. 235).

Figure 4, right, shows the Hasse diagram of lattice $\mathcal{L}$ in the example.

3.3 Insertions with bounded objective value

Let $T = \{w_K^{p_K} : K \in \mathcal{J}^-\}$ be a stable set of size $|\mathcal{J}^-|$ in the conflict graph $H = (W, Y)$, i.e. T corresponds to a feasible insertion. Its objective value $f(T) = f(\alpha(T))$ can be calculated in $(\mathcal{J}^+, B \cup U_R \cup U_T, c)$ as follows:

1. Determine the length $g(T)$ of a longest path from σ to J:
$$g(T) = \max\{l_{\sigma J};\, l_{\sigma K} + c_{KJ}^{p_K} : K \in \mathcal{J}^-\} \tag{9}$$
2. For each $K \in \mathcal{J}^- \cup \tau$, determine the length $h_K(T)$ of a longest path from J to K:
$$h_K(T) = \max\{c_{JL}^{p_L} + l_{LK} : L \in \mathcal{J}^- \cup \tau\} \tag{10}$$
3. For each $K \in \mathcal{J} \cup \tau$, determine the length $\alpha_K(T)$ of a longest path from σ to K, and therefore the earliest starting time of K by
$$\alpha_K(T) = \max\{l_{\sigma K}; g(T) + h_K(T)\} \tag{11}$$
with the convention $h_J(T) = 0$.
4. Compute $f(T) = f(\alpha(T))$.

Next, we examine structural properties of the family of feasible insertions of bounded objective value, motivated by results we obtained earlier on the job insertion problem in the NWJS with makespan objective. For any scalar ρ, denote by
$$\mathcal{L}^\rho = \{T \in \mathcal{L} : f(\alpha(T)) < \rho\}.$$

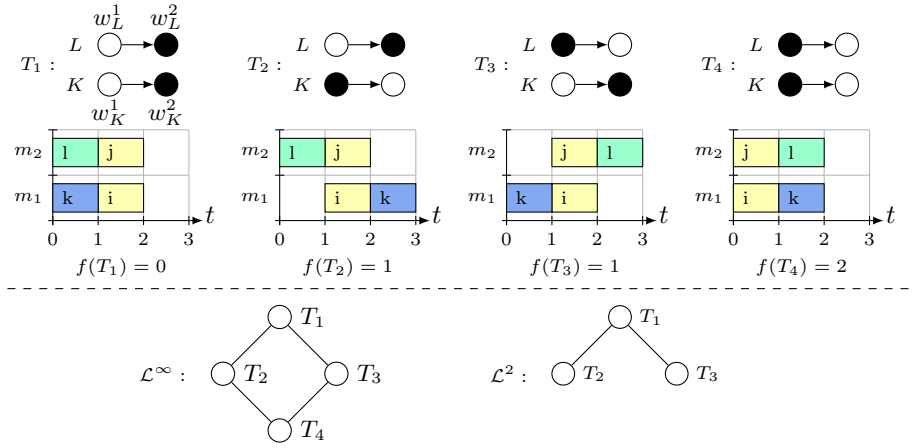

Figure 5: An example showing that $\mathcal{L}^\rho$ might not be a lattice.

3.3.1 The makespan case

In the special case where f is the makespan, i.e. $f(T) = \alpha_\tau(T)$, the family $\mathcal{L}^\rho$ has a nice characterization and played a key role in our optimal job insertion algorithm. We briefly recall without proofs some results of [2], assuming $\rho > l_{\sigma\tau}$.

Definition 8. *The conflict graph at ρ is the graph $H^\rho = (W^\rho, Y^\rho)$ with the node set W^ρ and the arc set Y^ρ defined by:*

$$\text{for all } p = 1, \dots, q_K \text{ and } K \in \mathcal{J}^-\text{:}$$
$$w_K^p \in W^\rho \Leftrightarrow \begin{cases} c_{\sigma J} + c_{JK}^p + l_{K\tau} < \rho \text{ and} \\ l_{\sigma K} + c_{KJ}^p + c_{J\tau} < \rho \end{cases} \tag{12}$$

$$\text{for all pairs of distinct nodes } w_K^p,\ w_L^q \in W^\rho\text{:}$$
$$(w_K^p, w_L^q) \in Y^\rho \Leftrightarrow \begin{cases} c_{JK}^p + c_{LJ}^q + l_{KL} > 0 \text{ or} \\ c_{JK}^p + c_{LJ}^q + l_{K\tau} + l_{\sigma L} \geq \rho \end{cases} \tag{13}$$

Theorem 9. *There is a 1 to 1-correspondence between the feasible insertions $T \in \mathcal{L}^\rho$ and the stable sets of size $|\mathcal{J}^-|$ in H^ρ.*

Theorem 10. *For any ρ, $H^\rho = (W^\rho, Y^\rho)$ is acyclic and transitively oriented.*

Proposition 11. *$\mathcal{L}^\rho$ is a lattice.*

Note that for sufficiently large ρ, H^ρ is the conflict graph H of the previous section and the three results above yield Theorems 5 and 6 and Proposition 7.

3.3.2 General regular function

For a general regular function f, $\mathcal{L}^\rho = \{T \in \mathcal{L} : f(T)) < \rho\}$ cannot be characterized similarly by stable sets in a conflict graph H^ρ. Indeed, $\mathcal{L}^\rho$ might not be a lattice as shown in the following example, which is illustrated in Figure 5.

Consider three jobs J, K, L and two machines m_1, m_2. J has two operations i on m_1 and j on m_2 with $\gamma_{ij} = 0$, i.e. J occupies simultaneously m_1 and m_2. K has one operation k on m_1 and L has one operation l on m_2. All operations i, j, k, l have duration 1, job J has due date 3 and jobs K and L have due date 1. The objective function f is the number of tardy jobs.

In the insertion problem for J, the conflict graph $H = (W, Y)$ has nodes w_K^1, w_K^2, w_L^1, w_L^2 and arcs (w_K^1, w_K^2), (w_L^1, w_L^2). As illustrated in Figure 5, there exists four feasible insertions. Consider $T_2 = \{w_K^1, w_L^2\}$, $T_3 = \{w_K^2, w_L^1\}$, and $T_2 \wedge T_3 = T_4 = \{w_K^1, w_L^1\}$, corresponding to inserting J respectively before K and after L, before L and after K, and before K and L. Then $f(T_2) = f(T_3) = 1$ and $f(T_2 \wedge T_3) = 2$, hence for $\rho = 2$, T_2 and $T_3 \in \mathcal{L}^\rho$, but $T_2 \wedge T_3 \notin \mathcal{L}^\rho$.

However, the following result holds.

Theorem 12. *$\mathcal{L}^\rho = \{T \in \mathcal{L} : f(T) < \rho\}$ is a $\vee$-semi-lattice.*

Proof. Let $T = \{w_K^{p_K} : K \in \mathcal{J}^-\}$ and $T' = \{w_K^{p'_K} : K \in \mathcal{J}^-\}$ be any members of $\mathcal{L}$.

First, we show that the function g defined in (9) satisfies

$$g(T \vee T') \leq \max\{g(T), g(T')\}. \tag{14}$$

$g(T \vee T') = \max\{l_{\sigma J};\ l_{\sigma K} + c_{KJ}^{p_K^\vee} : K \in \mathcal{J}^-\}$ where $p_K^\vee = \max\{p_K, p'_K\}$. If $g(T \vee T') = l_{\sigma J}$, clearly both $g(T \vee T') \leq g(T)$ and $g(T \vee T') \leq g(T')$. Assume $g(T \vee T') = l_{\sigma K^*} + c_{K^*J}^{p_{K^*}^\vee}$ for some $K^* \in \mathcal{J}^-$. If $p_{K^*}^\vee = p_{K^*}$, $g(T \vee T') = l_{\sigma K^*} + c_{K^*J}^{p_{K^*}} \leq g(T)$, and if $p_{K^*}^\vee = p'_{K^*}$, $g(T \vee T') \leq g(T')$.

Next, for any $K \in \mathcal{J} \cup \tau$, the function h_K defined in (10) satisfies

$$h_K(T) \leq h_K(T') \text{ if } T' \preceq T \tag{15}$$

Indeed, $T' \preceq T$ implies $p'_L \leq p_L$ for all $L \in \mathcal{J}^-$. Then, by Proposition 2, $c_{JL}^{p'_L} \geq c_{JL}^{p_L}$ for all $L \in \mathcal{J}^-$. Therefore $h_K(T) = \max\{c_{JL}^{p_L} + l_{LK} : L \in \mathcal{J}^-\} \leq h_K(T') = \max\{c_{JL}^{p'_L} + l_{LK} : L \in \mathcal{J}^-\}$ for any $K \in \mathcal{J}^- \cup \tau$.

Then,

$$f(T \vee T') \leq \max\{f(T), f(T')\}.$$

Indeed, in view of (14), we may assume without loss of generality $g(T \vee T') \leq g(T) = \max\{g(T), g(T')\}$. Also, by (15) $h_K(T \vee T') \leq h_K(T)$ holds for all $K \in \mathcal{J} \cup \tau$. Hence, using (11), $\alpha_K(T \vee T') = \max\{l_{\sigma K}; g(T \vee T') + h_K(T \vee T')\} \leq \max\{l_{\sigma K}, g(T) + h_K(T)\} = \alpha_K(T)$ for all $K \in \mathcal{J} \cup \tau$. Since f is regular, $f(\alpha(T \vee T')) \leq f(\alpha(T))$.

Finally, for any $\rho \in \mathbb{R}$ and any $T, T' \in \mathcal{L}^\rho$,

$$f(\alpha(T \vee T')) \leq \max\{f(\alpha(T)), f(\alpha(T'))\} < \rho$$

therefore $T \vee T' \in \mathcal{L}^\rho$, hence $\mathcal{L}^\rho$ is a $\vee$-semi-lattice. □

3.4 The OJI-NWJS-R algorithm

We propose an efficient method for the OJI-NWJS-R, the OJI-NWJS-R algorithm. We describe below its principle and then prove its validity.

Given any stable set $T = \{w_K^{p_K} : K \in \mathcal{J}^-\}$, in step 1 of the calculation of its objective value $f(T)$ detailed in the previous section, define

$$\begin{aligned} \mathcal{R}_T &= \{K \in \mathcal{J}^- : l_{\sigma K} + c_{KJ}^{p_K} = g(T)\}, \\ Q_T &= \{w_K^{p_K} : K \in \mathcal{R}_T\} \subseteq T \text{ and} \\ \mathcal{L}_T &= \{T' \in \mathcal{L} : T' \prec T \text{ and } T' \cap Q_T = \emptyset\}. \end{aligned} \tag{16}$$

Note that if $Q_T \neq \emptyset$, $\mathcal{L}_T$ is a sublattice of $\mathcal{L}$.

In the OJI-NWJS-R algorithm, a sequence $T_0, T_1, \ldots, T_s$ is generated where:

$$\begin{gathered} T_0 \text{ is the maximal element of } \mathcal{L}, \\ \text{for } r = 0, 1, \ldots, s-1,\ T_{r+1} \text{ is the maximal element of } \mathcal{L}_{T_r} \\ \text{and } \mathcal{L}_{T_s} = \emptyset. \end{gathered}$$

Theorem 13. *T_p with $f(T_p) = \min\{f(T_r) : 0 \leq r \leq s\}$ is an optimal insertion.*

Proof. i) We first show that for any $T, T' \in \mathcal{L}$:

$$T \succ T' \text{ and } f(T) > f(T') \Rightarrow T' \in \mathcal{L}_T. \tag{17}$$

Note that $Q_T \neq \emptyset$. Indeed, if $Q_T = \emptyset$, $\mathcal{R}_T = \emptyset$ and therefore $g(T) = l_{\sigma J} \leq g(T')$. Also, since $T \succ T'$, $h_K(T) \leq h_K(T')$ by (15) for all $K \in \mathcal{J} \cup \tau$, hence $f(T) \leq f(T')$, a contradiction.

Let $T = \{w_K^{p_K} : K \in \mathcal{J}^-\}$, $T' = \{w_K^{p'_K} : K \in \mathcal{J}^-\}$ and assume $T' \cap Q_T \neq \emptyset$. Then $p'_L = p_L$ for some $L \in \mathcal{R}_T$, therefore $g(T) = l_{\sigma L} + c_{LJ}^{p_L} \leq g(T') = \max\{l_{\sigma J}; l_{\sigma K} + c_{KJ}^{p'_K} : K \in \mathcal{J}^-\}$. The same argument as above then leads to contradicting $f(T') < f(T)$. Hence $T' \in \mathcal{L}_T$.

ii) Let T^* be an optimal insertion and suppose

$$f(T^*) < f(T_r) \text{ for all } r = 0, \ldots, s.$$

Clearly $T_0 \succ T_1 \succ \ldots \succ T_s$ and $T_0 \succ T^*$. Also, $T_s \not\succ T^*$. Indeed, if $T_s \succ T^*$, applying (17) for $T = T_s$ and $T' = T^*$ leads to the contradiction $T^* \in \mathcal{L}_{T_s} = \emptyset$. Therefore there exists a largest r such that $T_r \succ T^*$, and $r < s$. Applying (17) for $T = T_r$ and $T' = T^*$ implies $T^* \in \mathcal{L}_{T_r}$. But then, since T_{r+1} is the maximal element of $\mathcal{L}_{T_r}$, $T_{r+1} \succ T^*$ contradicting the choice of r. □

It remains to be shown, given r, $0 \leq r < s$, how to find the maximal element T_{r+1} of $\mathcal{L}_{T_r}$.

Assume $T_r = \{w_K^{p_K} : K \in \mathcal{J}^-\}$ and denote Q_{T_r} by Q_r. Let $H^r = (W^r, Y^r)$ be the subgraph obtained by deleting in H all nodes w_K^p, $p > p_K$, $K \in \mathcal{J}^-$.

For any $K \in \mathcal{J}^-$ and $v \in W_K^r = W_K \cap W^r$, denote by $p(v)$ the immediate predecessor of v in W_K^r, with the convention $p(v) = \emptyset$ if $v = w_K^1$. For any $v, w \in W^r$, define the relations $v \mapsto w$ if $(p(v), w) \in Y^r$, and $v \rightsquigarrow w$ if there is a sequence of distinct nodes $v_l \in W^r$, $l = 1, \ldots, q$, $q \geq 1$, such that $v_1 = v$, $v_q = w$ and $v_l \mapsto v_{l+1}$ for $l = 1, \ldots, q-1$. If $v \rightsquigarrow w$, the distance $\lambda(v, w)$ from v to w is given by the minimum length q of such a sequence. Define

$$\Psi^r(Q_r) = \{w \in W^r : v \rightsquigarrow w \text{ for some } v \in Q_r\}.$$

Lemma 14. *If $W^r_K - \Psi^r(Q_r) = \emptyset$ for some $K \in \mathcal{J}^-$, then $\mathcal{L}_{T_r} = \emptyset$, else the maximal element of $\mathcal{L}_{T_r}$ is $T_{r+1} = \{w^{p'_K}_K : K \in \mathcal{J}^-\}$, where $p'_K = \max\{p : w^p_K \in W^r_K - \Psi^r(Q_r)\}$.*

Proof. i) First, assuming $\mathcal{L}_{T_r} \neq \emptyset$, we show that for any $T' \in \mathcal{L}_{T_r}$, $T' \cap \Psi^r(Q_r) = \emptyset$. Suppose to the contrary some $T' \in \mathcal{L}_{T_r}$ with $T' \cap \Psi^r(Q_r) \neq \emptyset$. There exists $v \rightsquigarrow w$ for some $v \in Q_r$ and $w \in T' \cap (\Psi^r(Q_r) - Q_r)$. Choose such v and w with minimal distance $\lambda(v, w)$, and let $v_1, v_2, \ldots, v_q$, with $v_1 = v$, $v_q = w$, $v_l \mapsto v_{l+1}$ for $l = 1, \ldots, q-1$ and $q = \lambda(v, w)$. Clearly, $v_l \notin T'$ for all $l < q$, in particular $v_{q-1} \notin T'$.

Now, assuming $v_{q-1} \in W^r_K$, the unique node $u \in T' \cap W^r_K$ must precede v_{q-1} in W^r_K. Indeed, this holds if $q = 2$, as $v_{q-1} = v_1 = v \in Q_r$ and $T' \cap Q_r = \emptyset$. If $q > 2$, $v_{q-2} \mapsto v_{q-1}$ implies $(p(v_{q-2}), v_{q-1}) \in Y^r$ and, by transitivity, $(p(v_{q-2}), u)$ for any successor u of v_{q-1} in W^r_K. If some successor u were in T, then $v \rightsquigarrow u$ and $\lambda(v, u) < q$, contradicting the choice of v and w.

Consequently, since u precedes v_{q-1} in W^r_K and $(p(v_{q-1}), w) \in Y^r$, $(u, w) \in Y^r$ holds by transitivity, a contradiction to the stability of T'.

ii) If $W^r_K - \Psi^r(Q_r) = \emptyset$ for some $K \in \mathcal{J}^-$, there is obviously no complete T' with $T' \cap \Psi^r(Q_r) = \emptyset$, hence $\mathcal{L}_{T_r} = \emptyset$.

iii) T_{r+1} is stable. Suppose to the contrary some $v, w \in T_{r+1}$ with $(v, w) \in Y^r$. If $v \in T_{r+1} - T_r$, $v = p(u)$ for some $u \in \Psi^r(Q_r)$. But then $u \mapsto w$ and therefore $w \in \Psi^r(Q_r)$, contradicting $T_{r+1} \cap \Psi^r(Q_r) = \emptyset$. If $v \in T_{r+1} \cap T_r$, the stability of T_r in H^r is contradicted. Indeed, if $w \in T_{r+1} \cap T_r$, then $(v, w) \in Y^r$ obviously leads to the contradiction. If $w \in T_{r+1} - T_r$ and $w \in W^r_K$ for some K, there is some $u \in T_r \cap W^r_K$, hence $(w, u) \in Y^r$ and, by transitivity, $(v, u) \in Y^r$.

Therefore T_{r+1} is the maximal element of $\mathcal{L}_{T_r}$. □

The algorithm can be summarized as follows.

The OJI-NWJS-R algorithm

Initialization: $f^* := \infty$; $T := \{w^{q_K}_K : K \in \mathcal{J}^-\}$; *optimal* := $false$;
while *optimal* = $false$ **do**
  Determine $\alpha_K(T)$ for all $K \in \mathcal{J} \cup \tau$ and Q_T;
  Determine $f(T)$; **if** $f(T) < f^*$ **then** $f^* := f(T)$ and $T^* := T$; **end** (if)
  if $Q_T = \emptyset$ **then** *optimal* := *true*; **return**;
  else compute $\Psi(Q_T)$ in $H = (W, Y)$; **end** (if)
  if $W_K - \Psi(Q_T) = \emptyset$ for some $K \in \mathcal{J}^-$ **then** *optimal* := *true*; **return**;
  else $p_K := \max\{p : w^p_K \in W_K - \Psi(Q_T)\}$ for all $K \in \mathcal{J}^-$;
    $T := \{w^{p_K}_K : K \in \mathcal{J}^-\}$;
    Delete in H for all $K \in \mathcal{J}^-$ nodes w^p_K with $p > p_K$;
  end (if)
end (while)
T^* is an optimal insertion.

The number of iterations of the while loop is bounded by $\sum_{K \in \mathcal{J}^-} q_K$. Indeed, if $T_0, T_1, \ldots, T_s$ is the sequence of sets T generated by the algorithm, $T_0 \succ T_1 \succ \ldots \succ T_s$ so that $s \leq \sum_{K \in \mathcal{J}^-} q_K$. The complexity of the algorithm also depends on the effort for i) determining $\alpha_K(T)$ for all $K \in \mathcal{J} \cup \sigma$, Q_T and $f(T)$, and ii) computing $\Psi(Q_T)$. We give in the Appendix a detailed implementation that achieves the following computational complexity, where n denotes the number of jobs and η is the effort for evaluating the objective value $f(T)$.

objective function	T_0	T_1	T_2	T_3	T_4
a) makespan	10	**9**	10	11	10
b) total flow time	29	29	**28**	31	30
c) total squared flow time	229	227	**214**	273	242
d) maximum tardiness	5	4	**2**	3	**2**
e) total tardiness	5	5	**4**	6	**4**
f) total squared tardiness	25	17	8	14	**6**
g) number of tardy jobs	**1**	2	2	3	3

Table 1: Objective values of the insertions generated in the example run (bold: optimal values).

Theorem 15. *The OJI-NWJS-R algorithm runs in time*

$$\mathcal{O}(\max\{n^3, \max\{n, \eta\} \cdot \textstyle\sum_K q_K\}).$$

To illustrate this result, we remark that most often (for total flow time, total tardiness, number of tardy jobs, etc.) η is of the order $\mathcal{O}(n)$. Also, a job has usually at most one operation on a given machine and consequently $\sum_{K \in \mathcal{J}^-} q_K \leq (n-1)(m+1)$, m being the number of machines. The OJI-NWJS-R algorithm then runs in time $\mathcal{O}(n^2 \cdot \max\{m, n\})$.

Considering the job insertion problem of J in the example, a run of the OJI-NWJS-R algorithm is depicted in Figure 6. The topmost part illustrates the situation after the initialization. Set $T = T_0$ is depicted in conflict graph H (left) by the grey nodes. T is the maximal member (grey node) of lattice $\mathcal{L}$ (middle). Note that for each member of the lattice, its elements $\{w_K^{p_K}, w_L^{p_L}, w_N^{p_N}\}$ are simply described by (p_K, p_L, p_N). The schedule obtained with the earliest starting times $\alpha(T)$ is shown in a Gantt chart (right). Set Q_T consists of the nodes that are surrounded by a dashed box in H. For each iteration $r = 1, \ldots, 4$ of the while-loop the obtained set $T = T_r$ is illustrated in graph H^{r-1} (left), lattice $L_{T_{r-1}}$ (middle), and the corresponding schedule is on the right. Nodes that are not anymore part of the current conflict graph or lattice are depicted in light grey. The algorithm stops after four iterations as $Q_T = \emptyset$.

In Table 1, the objective value $f(T_r)$ of each set $T_r, r = 0, \ldots, 4$, is presented for different regular objectives f. For the objectives involving tardiness, we set the due dates of jobs (J, K, L, N) to $(5, 8, 8, 5)$. Optimal insertions are T_0 for objective g), T_1 for a), T_2 for b), c), d), e), and T_4 for f).

4 An optimal job insertion-based method for the NWJS-R

4.1 A simple local search

As in [2], we develop a heuristic for the NWJS-R that is based on the OJI-NWJS-R algorithm. It can be described as follows.

A feasible initial selection is constructed by successively inserting optimally a job. The initial selection is then improved by local search. We consider two neighborhoods. In neighborhood N^1, a neighbor is generated for each job by

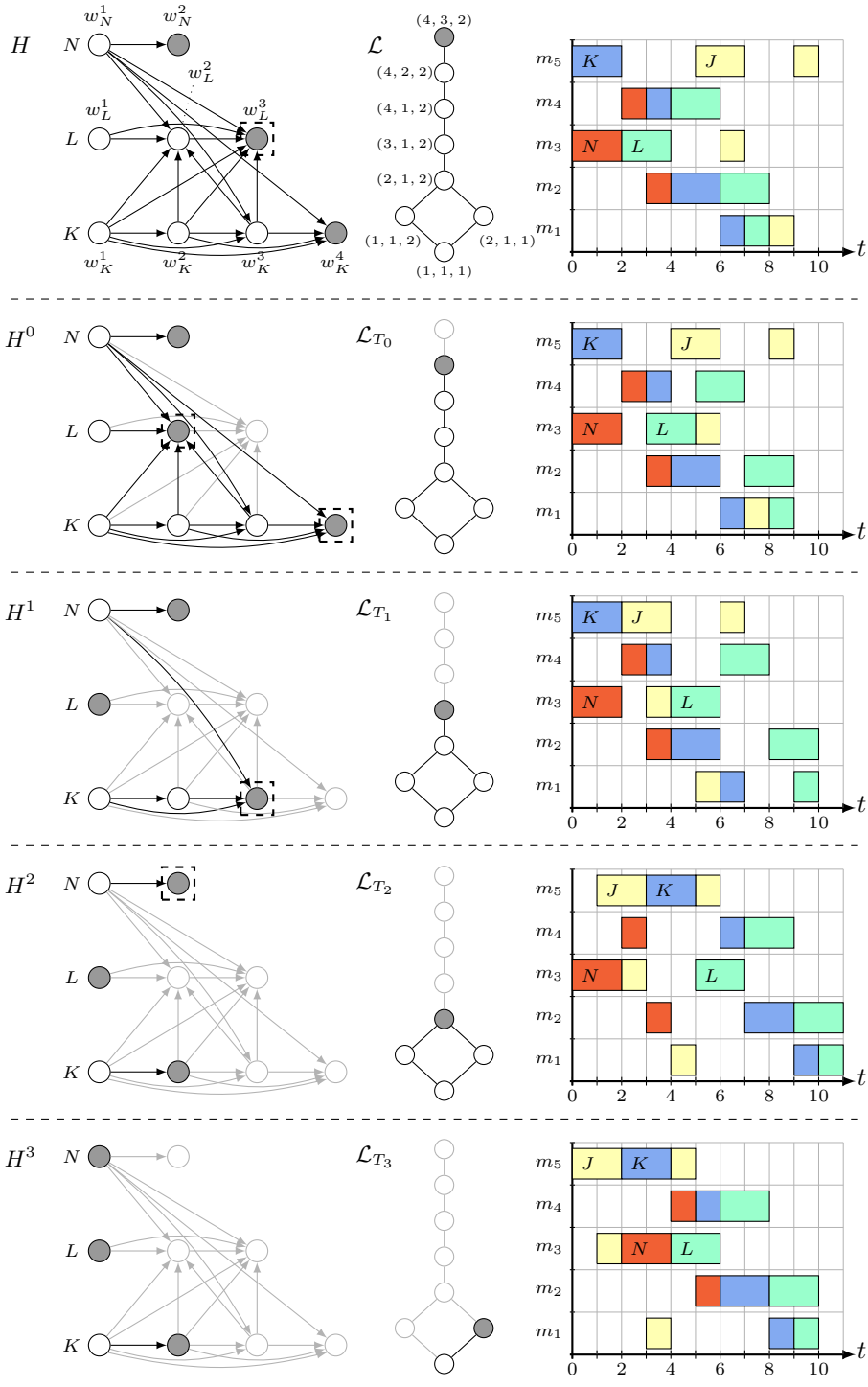

Figure 6: A run of the OJI-NWJS-R algorithm in the example.

extracting and reinserting optimally this job. Neighborhood N^1 has size $|J|$, which is quite small. Therefore, we examine a larger neighborhood N^2, where for any pair of ordered jobs, a neighbor is generated by extracting these two jobs and reinserting them successively. N^2 has size $|J|^2$.

Based on N^1 and N^2, two repeated simple descent local search methods are considered. In the first version, named OJIRLS1, we use neighborhood N^1 and select the first improvement, i.e. if a neighbor yields a lower objective value, the current selection is reset to this neighbor. In the second version, named OJIRLS2, we use neighborhood N^1 until a local optimal selection is found. Then, we continue with neighborhood N^2. If a neighbor in N^2 yields a lower objective value, the current selection is reset to this neighbor and OJIRLS2 continues again with neighborhood N^1. The algorithm stops if no better neighbor is found in N^2.

OJIRLS1 and OJIRLS2 are repeated from various initial selections generated by random job insertion orders until a given run time limit is reached.

4.2 Computational experiments

We implemented OJIRLS1 and OJIRLS2 in Java and run them on a PC with 3.4 GHz processor and 12 GB memory. Extensive tests were performed to evaluate the two methods. We used the well-known benchmark instances la11-15/26-35 proposed by Lawrence [4], swv06-20 by Storer et al. [10], and yn1-4 by Yamada and Nakano [11]. These instances were interpreted as "classical" no-wait job shop instances, i.e. $\gamma_{ij} = p_i$ for each pair i, j of consecutive operations of a job, and all set-up times and release times are 0. The following seven objective functions were considered: a) makespan, b) total flow time, c) total squared flow time, d) maximum tardiness, e) total tardiness, f) total squared tardiness, and g) number of tardy jobs. For the objectives d) to g), the due date d_J of each job $J \in \mathcal{J}$ is set according to the simple and popular rule introduced by Eilon and Hodgson [3] for the classical job shop: $d_J = \lfloor f * \sum_{i \in J} p_i \rfloor$, where f is referred to as the due date tightness factor. f is set to 3.0 for instances of size (20×15) and (20×20), 3.5 for (20×5) and (20×10), 5.0 for (30×10), and 7.0 for (50×10), where $(n \times m)$ refers to the number n of jobs and the number m of machines.

4.3 Quality of local optimal solutions

We first evaluated the quality of the local optimal solutions for both versions OJIRLS1 and OJIRLS2. For this purpose, we run OJIRLS1 and OJIRLS2 for each instance and objective function with a computation time limit of 7200 seconds. The objective value of each attained local optimal solution was recorded.

As an illustration, Figure 7 shows for the instance la31 and total flow time objective the sampling distribution of the local optimal values (in units of 1000) with OJIRLS1 (left) and OJIRLS2 (right). A box plot highlights the minimum value, lower quartile, median, upper quartile and maximum value.

We note that both the mean and the variance of the distribution are smaller for OJIRLS2 than for OJIRLS1. However, the average time needed to generate a local optimal solution with OJIRLS2 is much higher. Indeed, 1 088 070 local optimal solutions were found within 7200 seconds with OJIRLS1 versus 29 893

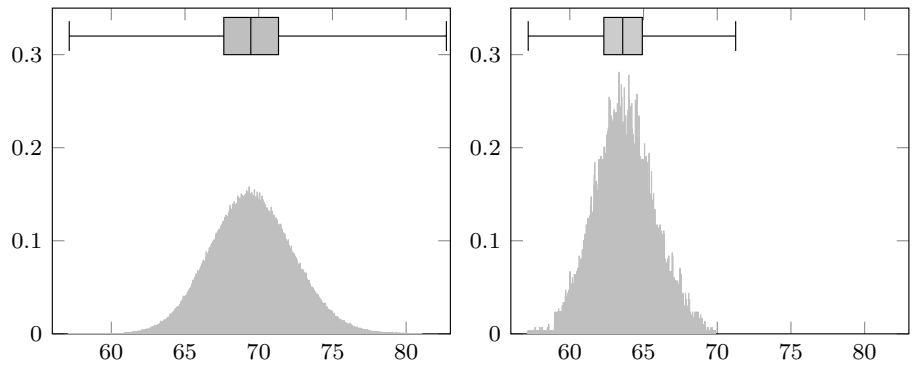

Figure 7: Distributions of the local optimal values (in units of 1000) for instance la31 and total flow time objective for OJIRLS1 (left) and OJIRLS2 (right).

solutions with OJIRLS2. These observations illustrate well the trade-off between solution quality and required computation time.

We examined the magnitude of this trade-off for each instance with the makespan and total flow time objective. Denote by f^1 and f^2 the average local optimal value obtained with OJIRLS1 and OJIRLS2, respectively, and similarly, let t^1 and t^2 be the average time needed per local search repetition, i.e. the number of obtained local optimal solutions divided by the total run time. Figure 8 illustrates the quality-time trade-off as follows. The horizontal and vertical axes depict the relative difference of the time, i.e. $(t^2 - t^1)/t^1$, and the relative difference of the solution quality, i.e. $(f^2 - f^1)/f^1$. Each combination of instance and objective is represented by a symbol. For example, the instance la31 with total flow time objective is represented by a grey square at $(35.4, -8.5\%)$. The same symbols are used for instances of the same size and objective.

We observe that the value of the local optimal solutions is between 5% to 9% lower with OJIRLS2, but its computation time is by a factor of 5 to 90 larger.

4.4 Performance of OJIRLS1 and OJIRLS2

It is of interest to evaluate the performance of OJIRLS1 and OJIRLS2 with a given run time limit. For this purpose, we considered the solution quality of OJIRLS1 and OJIRLS2 obtained after a short (60 seconds), medium (300 seconds), and high (1800 seconds) run time. For the evaluation of the average behavior (over multiple runs), we could in principle use the results of the previous experiment by dividing the single run of 7200 seconds into time slots of 60, 300, and 1800 seconds, respectively, determining the best solution obtained within each time slot, and presenting the average results.

As the number of local search repetitions executed within a given time is quite constant in a run, a better estimation of the average behavior is obtained with the following variance reduction technique, sometimes called simulation shortcut (see e.g. McGeoch [6]). For a given time limit, we determine the average number of local search repetitions r executable within that time limit, and then simulate a run by sampling randomly and independently r values from

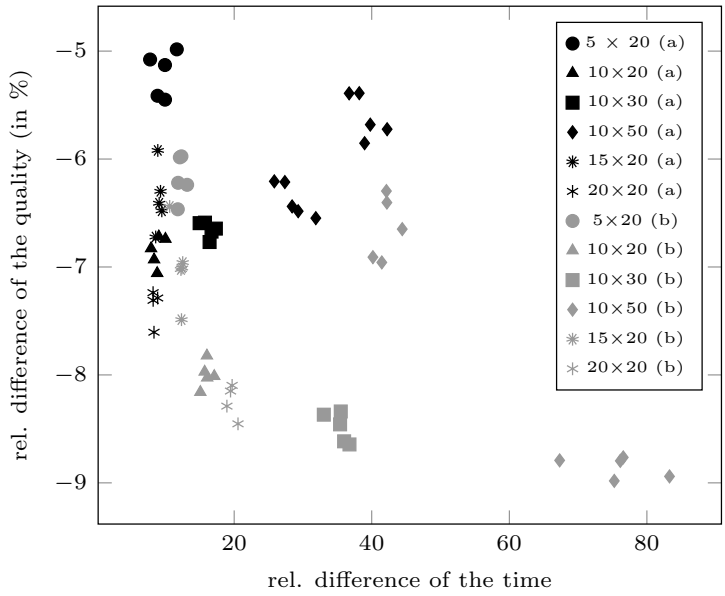

Figure 8: Quality-time trade-off in all instances with the makespan (a) and total flow time (b) objective.

the distribution obtained by the single run of 7200 seconds. The objective value of a run is then the minimum value of the r samples.

According to this procedure, we simulated 100 runs for the two methods OJIRLS1 and OJIRLS2, each instance, each objective function and the three run time limits (60, 300, and 1800 seconds). Detailed results can be found in Tables 7 to 9 in the Appendix. We now discuss these results.

It is of interest to examine the evolution of attained solution quality during computation. For this purpose, we determined the relative gap of the average results (over the 100 runs) after 60 and 300 seconds from the average results after 1800 seconds. Table 2 provides these numbers in an aggregated form (averaged over all instances of the same size) for objectives a) to c). In addition, columns 2 and 9 give the average number of local search repetitions obtained in 1800 seconds (in units of 1000, averaged over all instances of the same size and objectives a) to c)).

The following observations can be made. The values obtained after a short and medium run time are quite close to those obtained with a high run time. Indeed, after 60 and 300 seconds, the results are improved only by about 2.4% and 1.0%, respectively, in instances with the makespan and total flow time objective. These numbers are slightly higher for the total squared flow time objective. Both versions OJIRLS1 and OJIRLS2 behave similarly.

We then compared the results of OJIRLS1 and OJIRLS2 with each other. For each instance, objective function and run time limit, we determined the relative difference of the solution quality, i.e. $(s^2 - s^1)/s^1$, where s^1 and s^2 is the average value (over the 100 runs) obtained with OJIRLS1 and OJIRLS2, respectively. Table 3 reports these differences in an aggregated way (averaged over all instances of the same size) for objectives a) to c). The last line gives the average value over all instances.

method	OJIRLS1							OJIRLS2						
objective		makesp.		tot. f. t.		sq. f. t.			makesp.		tot. f. t.		sq. f. t.	
run time	rep.	60	300	60	300	60	300	rep.	60	300	60	300	60	300
20×5	697	2.4	1.2	2.0	0.9	4.0	1.9	63.4	2.1	0.9	1.0	0.2	3.3	1.2
20×10	737	3.6	1.8	2.3	1.1	5.1	2.3	60.7	3.2	1.4	1.5	0.5	3.4	1.3
30×10	291	2.6	1.3	2.8	1.2	6.0	2.9	9.7	2.7	1.3	2.8	1.3	6.0	2.9
50×10	38	2.0	1.0	2.5	1.3	4.6	2.2	0.8	2.3	1.0	2.7	1.3	5.0	2.2
20×15	639	1.6	0.5	1.7	0.7	3.5	1.6	56.8	0.7	0.1	0.7	0.1	1.3	0.1
20×20	714	3.8	1.9	3.4	1.6	7.3	3.2	53.5	3.5	1.6	3.0	1.2	6.5	2.9
all		2.5	1.2	2.4	1.1	5.0	2.3		2.4	1.0	2.0	0.8	4.3	1.8

Table 2: Relative gaps (in %) of the results after 60 and 300 seconds from the results after 1800 seconds, and average number of local search repetitions (in units of 1 000) for all instance sizes and objectives a) to c).

objective	makespan			total flow time			tot. squared flow t.		
run time	60	300	1800	60	300	1800	60	300	1800
20×5	-0.9%	-0.9%	-0.6%	-1.2%	-0.8%	-0.2%	-2.4%	-2.4%	-1.7%
20×10	-2.1%	-2.0%	-1.7%	-1.4%	-1.2%	-0.7%	-3.6%	-3.0%	-2.0%
30×10	-2.2%	-2.3%	-2.3%	-2.1%	-2.1%	-2.1%	-3.8%	-3.9%	-3.9%
50×10	-2.4%	-2.6%	-2.6%	-2.7%	-2.8%	-2.9%	-5.0%	-5.4%	-5.4%
20×15	-1.2%	-0.7%	-0.4%	-1.1%	-0.8%	-0.2%	-2.8%	-2.0%	-0.6%
20×20	-2.3%	-2.3%	-2.0%	-1.9%	-1.9%	-1.5%	-4.3%	-3.9%	-3.6%
all	-1.9%	-1.9%	-1.7%	-1.9%	-1.8%	-1.5%	-3.8%	-3.7%	-3.2%

Table 3: Relative differences of the results obtained with OJIRLS2 to those computed with OJIRLS1 for all instance sizes and objectives a) to c).

version	OJIRLS2	OJILS2	rel. diff.
20×5	308768	460680	-33.0%
20×10	321460	485886	-33.8%
30×10	55340	76859	-28.0%
50×10	4288	5924	-27.6%
20×15	276642	415739	-33.4%
20×20	293808	432429	-32.1%

run time	60	300	1800
20×5	0.3%	0.3%	0.1%
20×10	0.5%	0.4%	0.2%
30×10	0.4%	0.3%	0.2%
50×10	0.2%	0.2%	0.2%
20×15	0.2%	0.1%	0.0%
20×20	0.4%	0.3%	0.2%
all	0.3%	0.3%	0.2%

Table 4: (left) Number of local search repetitions executed in 7200 seconds with OJIRLS2 and OJILS2 and the makespan objective averaged over all instances of the same size and relative differences of these values. (right) Relative difference of the solution quality of OJIRLS2 compared to OJILS2 after 60, 300, and 1800 second averaged over all instances of the same size.

We observe that OJIRLS2 provides systematically substantially lower values than OJIRLS1. Indeed, the average values are up to 5.5% lower with OJIRLS2, and OJIRLS1 never provides a better value. Similar results were obtained for objectives d) to g). Altogether, the obtained results suggest that OJIRLS2 should be given preference over OJIRLS1.

4.5 Comparison with benchmarks

As is often the case in scheduling problems, an assessment of the obtained solution quality by comparing it with the proven optimum is not possible in the current state of the art. Also, benchmarks from the literature are only available for the NWJS with makespan objective. Therefore, we resorted to compare the performance of our approach to benchmark results in the makespan case and for the other objectives with results obtained via a mixed integer programming (MIP) model that we derived in a straightforward manner from the compact disjunctive graph formulation.

4.5.1 The makespan objective

As already mentioned, we developed a similar solution approach for the NWJS with makespan objective in our previous article [2]. In fact, OJIRLS1 and OJIRLS2 have a counterpart in [2], named OJILS1 and OJILS2. The only difference is the optimal job insertion algorithm that has been generalized in this work. With the makespan objective, the OJI-NWJS-R algorithm determines the *same* optimal job insertion as the algorithm developed in [2]. Hence, the same solutions are computed in OJIRLS1 and OJIRLS2 as in OJILS1 and OJILS2, respectively.

A difference is, however, the needed time to compute an optimal job insertion. To evaluate this difference, we run the (specialized) version OJILS2 from [2] under the same computational settings as OJIRLS2 with the same time limit of 7200 seconds, and compared the number of local search repetitions executed with the two methods. Columns 2 and 3 of Table 4 (left) provide these numbers in an aggregated form (averaged over all instances of the same size). Column 4 presents the relative difference of these values.

We observe that the number of local search repetitions is, on average, about 31% lower in the more general version OJIRLS2. Interestingly, the difference appears to be independent of the problem size.

To evaluate the effect of the increase in computation time on the solution quality, we determined the solution quality for OJILS2 in the same way as for OJIRLS2, namely by a simulation of 100 runs based on the single run of 7200 seconds. For each instance, we then determined the relative difference of the solution quality obtained by OJIRLS2 and OJILS2, i.e. $(s^r - s^n)/s^n$, where s^r and s^n is the average value (over the 100 runs) obtained with OJIRLS2 and OJILS2, respectively. Table 4 (right) presents these values for the three run time limits (60, 300, and 1800 seconds) averaged over all instances of the same size. The last line gives the average value over all instances.

We observe that the solution quality of OJIRLS2 is only marginally lower than OJILS2. As OJILS2 is currently among the best methods for the NWJS with makespan objective, we conclude that OJIRLS2 has a good performance with this objective.

In addition, it is well-known that any problem with maximum tardiness objective can be easily transformed into a makespan minimization problem. Hence, we conclude that OJIRLS2 performs also well with the maximum tardiness objective.

4.5.2 Other objective functions

Since no benchmark results are available for the other objectives, we tried to assess the quality of OJIRLS2 with results obtained via a MIP model (with linear or quadratic objective) using the solver Gurobi 6.0.

Preliminary tests revealed that solutions could only be found in small instances with the MIP approach. Therefore, we decided to use the instances la01-15/26-30 for these experiments. For the objectives e) to g), we set the tightness factor f to 2.0 for la01-10 and 3.5 for la11-15/26-30. For each instance and objective, the MIP model was run with a time limit of five hours. The objective values of the best solutions (i.e. the upper bound) and the lower bounds were recorded and compared to the average results (over the 100 runs) obtained with OJIRLS2 after 300 seconds. The detailed results can be found in Table 10 in the Appendix. Table 5 summarizes these results by presenting the relative difference of the average results (over the 100 runs) obtained with OJIRLS2 after 300 seconds (avg-300) and the MIP upper bound (UB), i.e. (avg-300 − UB) / UB (averaged over all instances of the same size).

The following observations can be made. Both OJIRLS2 and the MIP approach perform well in very small instances. Indeed, optimality was proven with the MIP approach for all instances of size (10×5) and all objectives after a few seconds, and OJIRLS2 always found an optimal solution. This was also the case for instances of size (15×5) with the number of tardy jobs objective.

In all other cases, OJIRLS2 gave substantially better results than the MIP approach. Also, the larger the instances are, the larger the performance difference is. Indeed, OJIRLS2 gives on average 5.3%, 25.4%, and 48.4% lower values than the MIP approach in instances of size (15×5), (20×5), and (20×10), respectively.

Altogether, the obtained results suggest that OJIRLS2 performs well with all considered objectives.

objective	total flow time	total squared flow time	total tardiness	total squared tardiness	number of tardy jobs	avg
10×5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15×5	-2.5%	-7.4%	-4.3%	-12.5%	0.0%	-5.3%
20×5	-8.0%	-22.0%	-25.3%	-64.7%	-6.8%	-25.4%
20×10	-18.3%	-36.5%	-66.6%	-90.4%	-30.3%	-48.4%

Table 5: Relative difference of the results obtained with OJIRLS2 after 300 seconds and the MIP results for objectives b), c), and e) to g) averaged over all instances of the same size. The last column depicts the difference averaged over all instances of the same size and all objectives.

4.6 Solutions obtained with different objectives

We conclude this section with a comparison of the best solutions of instance la12 obtained with the objectives d) maximum tardiness, e) total tardiness, f) total squared tardiness, and g) number of tardy jobs. Figure 9 illustrates these solutions in Gantt charts. Table 6 gives a list of all tardy jobs and presents the objective value for objectives d) to g) in each solution.

The following observations can be made. First, the solution found with the maximum tardiness objective (sol. d)) has quite high values for all other objectives. This is not surprising as in contrast to the others the maximum tardiness objective is not a min-sum objective. Second, a quite particular solution was obtained with the number of tardy jobs objective (sol. g)). Just five jobs are late, but these were scheduled one after the other at the end of the schedule. This might be inadequate in practice as the resulting tardiness of the tardy jobs is high. Third, in the solution found with the total squared tardiness objective (sol. f)), quite a lot of jobs are tardy but no job has a high tardiness. Therefore, the solution has also good values for the maximum and total tardiness objectives, but performs poorly when looking at the number of tardy jobs.

In practice, multiple objectives might be combined with different weights. As long as the objective function is regular, our approach can be applied.

5 Concluding remarks

We addressed the optimal job insertion problem in the no-wait job shop problem with general regular objective (NWJS-R) and developed a very efficient algorithm for this problem generalizing our results in case of a makespan objective.

Based on this algorithm, we proposed a simple local search for the NWJS-R and conducted computational experiments on a large set of instances and objectives. The obtained results compare favorably with current benchmarks, when available, and may serve as benchmarks for the community.

As future direction of research, it would be interesting to develop approaches for other complex job shop scheduling problems with a general regular objective. The obtained results suggest that local search methods based on optimal or "near-optimal" job insertion might be rewarding.

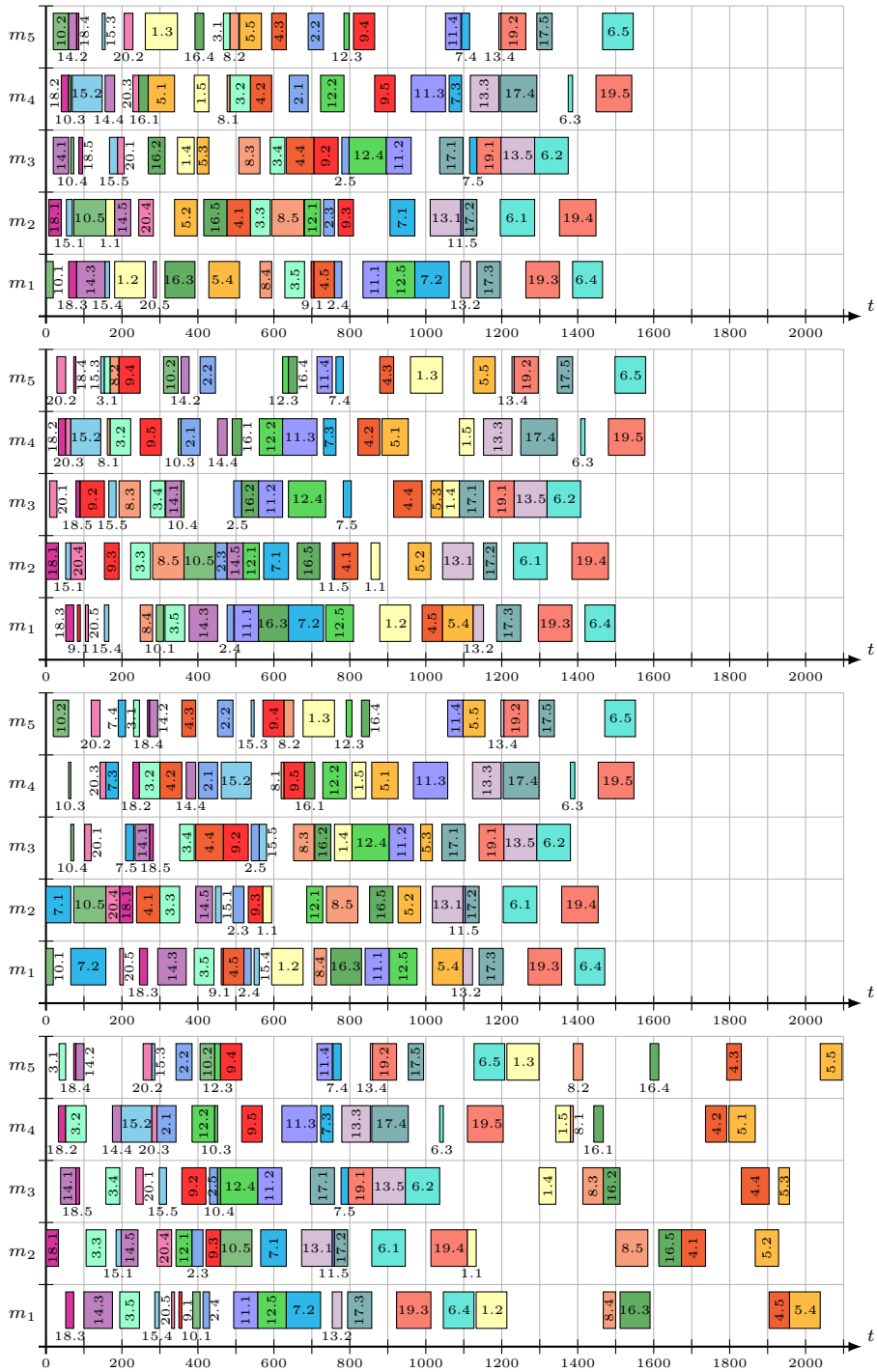

Figure 9: Four solutions of instance la12 obtained with the objectives (from top to bottom): d) maximum tardiness, e) total tardiness, f) total squared tardiness, and g) number of tardy jobs.

sol. d)		sol. e)		sol. f)		sol. g)	
job	tard.	job	tard.	job	tard.	job	tard.
7	330	13	360	13	333	5	1057
13	327	6	358	6	331	4	964
6	325	17	347	17	300	8	874
17	294	1	175	11	180	16	862
2	243	19	148	19	121	1	429
11	174	5	143	5	116		
9	149	4	50	8	112		
19	115			15	112		
				16	101		
				2	5		

objective	sol. d)	sol. e)	sol. f)	sol. g)	min.
max. tardiness	330 (0%)	360 (9%)	333 (1%)	1057 (220%)	330
tot. tardiness	1957 (24%)	1581 (0%)	1711 (8%)	4186 (165%)	1581
tot. squared tard. (in units of 1000)	533 (31%)	454 (12%)	406 (0%)	3738 (820%)	406
no. of tardy jobs	8 (60%)	7 (40%)	10 (100%)	5 (0%)	5

Table 6: (upper part) For each solution, a list of all tardy jobs indicating the job's number and the tardiness. (lower part) The objective value with objectives d) to g) in all solutions. The last column indicates the minimum objective value among all solutions. The values in brackets give the relative difference to the corresponding minimum value.

6 Appendix

A detailed implementation of the OJI-NWJS-R algorithm is given in Listing 1. The lines of the pseudo-code are numbered for reference in the following discussion.

```
1  // Initialization
2  Compute l_KL for all K,L ∈ J+; opt:=false;
3  for_all K ∈ J− ∪ τ do h_K := −∞; end_for
4  h_J := 0; f* := ∞; Δ := J−;
5  for_all K ∈ J− do p_K := q_K; end_for
6
7  while opt = false do
8    // Calculate earliest starting times of T := {w_K^{p_K} : K ∈ J−}.
9    g := max{l_σJ; l_σK + c_KJ^{p_K} : K ∈ J−};
10   if g > l_σJ then R := {K ∈ J− : l_σK + c_KJ^{p_K} = g}; end_if
11   for_all K ∈ J− do
12     h_K := max{h_K; c_JL^{p_L} + l_LK : L ∈ Δ};
13   end_for
14   h_τ := max{h_τ; l_Jτ; c_JL^{p_L} + l_Lτ : L ∈ Δ};
15   for_all K ∈ J ∪ τ do α_K := max{l_σK; g + h_K}; end_for
16   Evaluate f(α);
17   if f(α) < f* then
18     T* := {w_K^{p_K} : K ∈ J−}; f* := f(α);
19   end_if
20
21   if g = l_σJ then opt:=true; return; end_if
22   for_all K ∈ J− − R do p'_K := p_K; end_for
23   for_all K ∈ R do
24     if p_K = 1 then opt:=true; return; else
25       p'_K := p_K − 1; end_if
```

```
26    end_for
27    S := R;
28    while S ≠ ∅ do
29       get some K ∈ S;
30       for_all L ∈ J⁻ − K do
31          p := p'_L;
32          while (w_K^{p'_K}, w_L^p) ∈ Y do
33             if p = 1 then opt:=true; return; else
34                p := p − 1; end_if
35          end_while
36          if p < p'_L then p'_L := p; S := S ∪ L; end_if
37       end_for
38       S := S − K;
39    end_while
40
41    Δ := ∅;
42    for_all K ∈ J⁻ do
43       if p'_K < p_K then Δ := Δ ∪ K; p_K := p'_K; end_if
44    end_for
45 end_while
46 // T* is an optimal job insertion.
```

Listing 1: OJI-NWJS-R algorithm

Each iteration of the while loop (line 7) starts with the current insertion $T = \{w_K^{p_K} : K \in \mathcal{J}^-\}$, of which in a first part (lines 8 to 19), the earliest starting times α_K, $K \in \mathcal{J} \cup \tau$ and the objective value $f(\alpha)$ are computed, and the best insertion T^* is possibly updated. Then, in a second part (lines 21 to 39), the insertion T', maximal element of the lattice $\mathcal{L}_T$ as defined in (16) is computed or $\mathcal{L}_T = \emptyset$ is asserted (case opt=true). At completion of the iteration, $T' = \{w_K^{p'_K} : K \in \mathcal{J}^-\}$.

In order to detemine the computational complexity of the implementation, we first remark on the calculation of the earliest starting times α_K, $K \in \mathcal{J} \cup \tau$. For each $K \in \mathcal{J} \cup \tau$, h_K is updated (and not recalculated according to (10) in each iteration), based on the observation that h_K is monotone non-decreasing in the sequence of the iterations (see property (15)). Specifically, h_K in lines 12 and 14 is updated by considering its previous value and only those terms $c_{JL}^{p_L} + l_{LK}$ that have (potentially) changed since the previous iteration, i.e. terms $c_{JL}^{p_L} + l_{LK}$ for $K \in \Delta$, where Δ is computed in lines 41 to 45. It is then easy to show that the computational effort over all iterations is $\mathcal{O}(n \cdot \sum q_K)$ for calculating all α_K, $K \in \mathcal{J} \cup \tau$, and $\mathcal{O}(\eta \cdot \sum q_K)$ for evaluating f. The overall complexity of the first part is then $\mathcal{O}(\max\{n, \eta\} \cdot \sum q_K)$.

The overall complexity $\mathcal{O}(n \cdot \sum q_K)$ of the second part is achieved by growing $\Psi(Q_T)$ with a scanning phase and maintaining $T' = \{w_K^{p'_K} : K \in \mathcal{J}^-\}$ such that $T' \cap \Psi(Q_T) = \emptyset$ and for each $K \in \mathcal{J}^-$, $w_K^{p'_K} \in T$ or $w_K^{p'_K+1} \in \Psi(Q_T)$. This part is very similar to the scanning phase implemented in [2] and we do not expand on its details for this reason. Also, its overall complexity of $\mathcal{O}(n \cdot \sum q_K)$ can be shown by replicating arguments used in [2].

Altogether, taking into account the all-pairs longest paths computations to determine the l_{KL} (line 2), $K, L \in J^+$, the OJI-NWJS-R algorithm runs in time $O(n^3, \max\{n, \eta\} \cdot \sum q_K)$.

objective	makespan			total flow time			tot. squared flow time (in units of 10 000)		
run time	60	300	1800	60	300	1800	60	300	1800
20×5									
la11	1652	1636	1623	17256	17098	16980	1984	1942	1905
la12	1473	1458	1436	15135	14836	14644	1581	1554	1528
la13	1621	1603	1589	17068	16935	16771	1905	1864	1838
la14	1666	1637	1601	17918	17706	17520	2103	2049	1989
la15	1700	1681	1674	18229	18100	18036	2159	2121	2092
20×10									
la26	2604	2553	2510	29316	28938	28532	5384	5241	5123
la27	2759	2712	2673	30903	30530	30191	5986	5841	5723
la28	2663	2616	2580	29768	29472	29258	5558	5408	5291
la29	2456	2419	2374	27543	27249	27002	4739	4602	4514
la30	2628	2579	2515	28686	28321	27987	5313	5175	5021
30×10									
la31	3739	3691	3649	59636	58644	57815	15162	14657	14262
la32	4086	4033	3959	64777	63741	62944	17948	17473	17040
la33	3710	3663	3620	58998	58180	57391	14757	14353	13933
la34	3793	3750	3707	60826	59859	59227	15603	15146	14710
la35	3818	3765	3717	61207	60352	59699	15836	15415	14903
50×10									
swv11	5745	5703	5668	147388	145919	144425	56666	55503	54411
swv12	5773	5713	5653	148990	146810	144656	57551	56467	55565
swv13	5899	5840	5801	151643	150378	148988	60136	59088	58060
swv14	5649	5589	5519	145309	143885	142686	54894	53844	53090
swv15	5608	5563	5523	145088	143752	142519	54764	53731	52686
swv16	6195	6125	6057	158205	155871	153904	65708	63688	61924
swv17	6000	5924	5845	152096	150199	146951	61143	59354	57464
swv18	6037	5972	5916	154688	152871	151199	62832	61159	59473
swv19	6298	6234	6162	160329	158492	156229	67413	65885	64543
swv20	6003	5945	5889	153797	151444	149284	61899	60466	59021
20×15									
swv06	3326	3294	3286	38785	38557	38291	9135	9006	8930
swv07	3260	3226	3205	37612	37241	36939	8639	8486	8300
swv08	3478	3431	3423	40837	40046	39541	9925	9634	9359
swv09	3311	3279	3251	39084	38877	38716	9160	9031	8937
swv10	3530	3505	3482	40518	40261	40139	9870	9713	9617
20×20									
yn1	2503	2456	2409	28558	28139	27788	5022	4866	4759
yn2	2492	2445	2399	28811	28251	27790	5068	4887	4733
yn3	2450	2398	2349	28273	27686	27145	4871	4592	4420
yn4	2577	2535	2499	29681	29221	28791	5364	5216	5042

Table 7: Average results (over the 100 runs) of OJIRLS1 with objectives a) to c) and run time limits 60, 300, and 1800 seconds.

objective	makespan			total flow time			tot. squared flow time (in units of 10 000)		
run time	60	300	1800	60	300	1800	60	300	1800
20×5									
la11	1638	1624	1619	17090	16985	16980	1949	1925	1898
la12	1457	1439	1421	14890	14685	14631	1539	1484	1452
la13	1609	1595	1585	16888	16766	16741	1863	1837	1830
la14	1638	1607	1580	17650	17533	17473	2024	1973	1953
la15	1694	1679	1672	18075	17993	17933	2126	2096	2078
20×10									
la26	2547	2501	2480	28992	28675	28429	5212	5092	4999
la27	2717	2673	2630	30394	30141	29845	5732	5627	5533
la28	2615	2577	2554	29434	29224	29161	5379	5289	5246
la29	2397	2363	2323	26999	26682	26626	4531	4439	4389
la30	2566	2508	2455	28294	28016	27944	5151	5041	4983
30×10									
la31	3655	3602	3560	58470	57773	57317	14458	13995	13627
la32	3990	3931	3886	63387	62380	61584	17305	16745	16180
la33	3620	3575	3543	57606	56645	55809	14177	13831	13487
la34	3707	3660	3616	59602	58578	57699	15012	14571	14044
la35	3752	3696	3629	59912	59169	58353	15326	14884	14589
50×10									
swv11	5622	5566	5505	143652	141879	140683	54213	52636	51596
swv12	5649	5582	5535	145481	143343	141845	55118	53583	52194
swv13	5766	5688	5637	147785	146305	144614	56979	55792	54966
swv14	5528	5461	5388	141453	139977	137939	52306	50879	50014
swv15	5493	5431	5380	141837	140054	138815	52339	51390	50343
swv16	6035	5965	5897	153745	151530	149701	62247	60489	58976
swv17	5823	5759	5696	147834	145564	143528	57735	56325	55009
swv18	5893	5801	5740	150083	147529	145483	59090	57524	56348
swv19	6125	6038	5965	155448	153586	151551	63841	62083	60715
swv20	5867	5791	5749	149764	147248	143831	58650	56741	55121
20×15									
swv06	3296	3282	3278	38503	38278	38269	8991	8920	8915
swv07	3219	3196	3188	37218	36894	36880	8380	8278	8248
swv08	3437	3423	3423	40173	39631	39485	9440	9225	9221
swv09	3276	3251	3246	38727	38678	38669	8970	8894	8871
swv10	3482	3461	3453	40054	39991	39991	9649	9608	9607
20×20									
yn1	2450	2399	2369	28111	27786	27542	4839	4685	4589
yn2	2425	2376	2342	28227	27641	27367	4858	4669	4513
yn3	2387	2354	2317	27674	27087	26722	4605	4440	4298
yn4	2528	2482	2435	29123	28682	28255	5151	4999	4864

Table 8: Average results (over the 100 runs) of OJIRLS2 with objectives a) to c) and run time limits 60, 300, and 1800 seconds.

objective	max. tardiness			tot. tardiness			tot. squared tard. (in units of 1 000)			no. of tardy j.		
run time	60	300	1800	60	300	1800	60	300	1800	60	300	1800
20×5												
la11	385	381	381	1847	1843	1843	616	616	616	4.9	4.3	4.0
la12	335	332	330	1588	1581	1581	437	417	406	5.0	5.0	5.0
la13	395	392	392	1890	1820	1808	584	566	566	6.0	6.0	6.0
la14	442	432	428	2040	2009	2009	826	796	781	5.9	5.6	5.0
la15	483	475	474	2396	2364	2363	931	895	892	6.0	6.0	6.0
20×10												
la26	372	350	341	1473	1363	1324	485	416	394	3.7	3.1	3.0
la27	612	582	562	2328	2191	2094	1165	1005	884	5.0	4.9	4.6
la28	413	379	356	1575	1403	1356	547	418	368	3.8	3.4	3.0
la29	324	298	295	1307	1191	1114	366	330	325	3.6	3.1	3.0
la30	358	344	339	936	815	781	301	258	242	3.2	3.0	3.0
30×10												
la31	575	538	486	2871	2397	1853	1408	1125	998	5.7	5.2	4.9
la32	721	666	603	4084	3630	3267	2425	1874	1534	6.4	5.9	5.4
la33	562	506	443	3214	2875	2524	1629	1325	1140	5.7	5.1	4.8
la34	751	710	666	4192	3799	3389	2625	2178	1813	7.0	6.7	6.1
la35	749	703	634	3879	3532	3128	2373	1983	1605	6.9	6.4	6.0
50×10												
swv11	1032	974	930	10136	9167	8322	8234	7275	6704	12.3	11.5	11.0
swv12	1139	1064	1016	12186	11030	9604	11356	9878	9070	13.2	12.5	12.1
swv13	1099	1041	997	11175	10216	9427	9438	8175	7161	12.4	11.7	11.1
swv14	1086	1016	967	10475	9418	8280	8446	7262	6411	12.7	11.8	11.2
swv15	1155	1089	1030	11701	10716	10059	10824	9520	8634	13.3	12.6	12.1
swv16	1751	1681	1600	18709	17149	15906	25606	22542	18833	15.8	15.1	14.4
swv17	1546	1469	1421	17779	16203	14922	21823	19085	17234	14.5	13.8	13.2
swv18	1675	1585	1502	18545	16564	14736	23899	20582	18511	15.4	14.5	13.7
swv19	1774	1690	1601	20032	18532	17474	27367	23959	21310	15.9	15.3	14.7
swv20	1649	1562	1508	18607	17124	16038	24513	21206	18877	15.4	14.7	14.2
20×15												
swv06	645	644	644	2813	2810	2810	1321	1294	1294	4.9	4.6	4.1
swv07	696	689	689	2753	2735	2735	1408	1398	1398	5.1	5.0	5.0
swv08	748	746	746	3560	3528	3525	2055	2042	2036	5.0	4.9	4.4
swv09	771	750	748	3037	3005	3004	1933	1893	1893	5.4	5.0	5.0
swv10	862	854	854	3992	3964	3958	2545	2456	2445	6.0	5.9	5.7
20×20												
yn1	583	541	510	2186	1953	1767	1062	862	742	5.2	5.0	4.8
yn2	573	532	496	2254	1972	1759	1059	855	672	5.1	4.9	4.5
yn3	547	504	467	2101	1871	1634	921	752	631	5.3	5.0	5.0
yn4	520	489	456	2116	1889	1597	983	831	731	4.9	4.6	4.1

Table 9: Average results (over the 100 runs) of OJIRLS2 with objectives d) to g) and run time limits 60, 300, and 1800 seconds.

	total flow time			total squared flow time (in units of 10 000)			total tardiness			total squared tardiness (in units of 1 000)			number of tardy jobs		
	avg-300	UB	LB	avg-300	UB	LB	avg-300	UB	LB	avg-300	UB	LB	avg-300	UB	LB
10×5															
la01	5895	5895	(5 s.)	426	426	(12 s.)	962	962	(4 s.)	257	257	(13 s.)	4	4	(1 s.)
la02	5320	5320	(5 s.)	352	352	(20 s.)	606	606	(1 s.)	111	111	(13 s.)	3	3	(1 s.)
la03	5025	5025	(9 s.)	309	309	(55 s.)	805	805	(5 s.)	179	179	(15 s.)	4	4	(1 s.)
la04	5160	5160	(7 s.)	329	329	(21 s.)	864	864	(4 s.)	219	219	(66 s.)	4	4	(1 s.)
la05	4982	4982	(17 s.)	286	286	(54 s.)	987	987	(6 s.)	222	222	(41 s.)	4	4	(1 s.)
15×5															
la06	10636	11118	7512	967	1090	341	3502	3559	1488	1637	1974	60	8	8	(29 s.)
la07	10106	10449	7682	842	885	363	3304	3585	1477	1393	1703	75	8	8	(90 s.)
la08	10390	10679	7678	893	963	379	3355	3784	1504	1660	1927	57	8	8	(22 s.)
la09	11215	11458	8013	1076	1210	420	3448	3448	1413	1783	2055	140	8	8	(52 s.)
la10	10320	10320	8351	938	961	395	3035	3056	1067	1397	1397	43	8	8	(87 s.)
20×5															
la11	16985	17943	9203	1925	2523	372	1843	2091	0	616	1943	0	4	5	0
la12	14685	16234	7375	1484	1830	297	1581	2348	0	417	749	0	5	5	0
la13	16766	18277	8542	1837	2327	369	1820	2521	0	566	1750	0	6	6	0
la14	17533	19824	8970	1973	2602	345	2009	3153	0	796	3108	0	6	6	0
la15	17993	18972	9613	2096	2690	448	2364	2887	0	895	2891	0	6	7	0
20×10															
la26	28675	36230	15425	5092	8566	1099	1363	3505	0	416	5166	0	3	5	0
la27	30141	36848	16185	5627	9613	1164	2191	5050	0	1005	5351	0	5	6	0
la28	29224	37444	15375	5289	9382	1171	1403	4574	0	418	4506	0	3	5	0
la29	26682	32474	14014	4439	6249	998	1191	3370	0	330	6700	0	3	5	0
la30	28016	32076	15225	5041	6968	1404	815	4322	0	258	3699	0	3	4	0

Table 10: Average results of OJIRLS2 after 300 seconds (avg-300), and upper bounds (UB) and lower bound (LB) obtained with the MIP approach for objectives b), c), and e) to g). The run time is indicated (in seconds) in column LB if optimality could be proven within the time limit of five hours.

References

[1] P. Brucker and S. Knust. *Complex Scheduling*. Springer, second edition, 2011.

[2] R. Bürgy and H. Gröflin. Optimal job insertion in the no-wait job shop. *Journal of Combinatorial Optimization*, 26(2):345–371, 2013.

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

[4] S. Lawrence. Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. *GSIA, Carnegie Mellon University, Pittsburgh, PA*, 1984.

[5] 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, July 2011.

[6] C. McGeoch. Analyzing algorithms by simulation: variance reduction techniques and simulation speedups. *ACM Computing Surveys*, 24(2):195–212, 1992.

[7] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, fourth edition, 2012.

[8] A. Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency*. Springer, 2003.

[9] C. Schuster. No-wait job shop scheduling: tabu search and complexity of subproblems. *Mathematical Methods of Operations Research*, 63(3):473–491, 2006.

[10] R. H. Storer, S. D. Wu, and R. Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38(10):1495–1509, Oct. 1992.

[11] T. Yamada and R. Nakano. A genetic algorithm applicable to large-scale job-shop problems. *Parallel Problem Solving from Nature*, 2:281–290, 1992.