*Int. J. Appl. Math. Comput. Sci., 2024, Vol. 34, No. 2, 235–251*
DOI: 10.61822/amcs-2024-0017

# NEW IDEAS IN LAGRANGIAN RELAXATION FOR A SCHEDULING PROBLEM WITH THE WEIGHTED TARDINESS CRITERION

JAROSŁAW RUDY [a,*], RADOSŁAW IDZIKOWSKI [a], CZESŁAW SMUTNICKI [b], ZBIGNIEW BANASZAK [c], GRZEGORZ BOCEWICZ [c]

[a]Department of Control Systems and Mechatronics
Wrocław University of Science and Technology
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
e-mail: {jaroslaw.rudy,radoslaw.idzikowski}@pwr.edu.pl

[b]Department of Computer Engineering
Wrocław University of Science and Technology
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
e-mail: czeslaw.smutnicki@pwr.edu.pl

[c]Department of Computer Science and Management
Koszalin University of Technology
Śniadeckich 2, 75-453 Koszalin, Poland
e-mail: {zbigniew.banaszak,grzegorz.bocewicz}@tu.koszalin.pl

We consider an extension of Lagrangian relaxation methods for solving the total weighted tardiness scheduling problem on a single machine. First, we investigate a straightforward relaxation method and decompose it into upper and lower subproblems. For the upper subproblem we propose an alternative solving method in the form of a local search metaheuristic. We also introduce a scaling technique by arbitrary numbers to reduce the complexity of the problem and confront it with greatest common divisor scaling. Next, we propose a novel alternative relaxation approach based on aggregating constraints. We discuss the properties and implementation of this new approach and a technique to further reduce its computational complexity. We perform a number of computer experiments on instances based on the OR-Library generation scheme to illustrate and ascertain the numerical properties of the proposed methods. The results indicate that for larger instances the proposed alternative relaxation and scaling approaches have a much better convergence rate with little to no decrease in solution quality. The results also show that the proposed local-search metaheuristic is a viable alternative to the existing solving methods.

**Keywords:** Lagrangian relaxation, scheduling, total weighted tardiness, metaheuristics.

## 1. Introduction

Meeting customer due dates is an important issue of industrial operations. It refers to the manufacturing sector as well as to the service sector. It remains a central issue from any perspective within a network of operations, whether it be an individual company, a department within a company, or an entire collection of companies comprising a supply chain. Manufacturing companies that organize the flow of materials through work centers need to decide the sequence in which orders are to be processed at each work center. The choice of the sequence is heavily influenced by the desire to meet customer due dates, making minimization of tardiness an important goal (Liu *et al.*, 2023). Often the manufacturer assigns different priorities to orders based on specific customers. The problem of minimizing the total tardiness is the most commonly considered case. With regard to the total weighted tardiness scheduling problem and its relaxations, the paper contains the following contributions:

*Corresponding author

1. We propose a local-search metaheuristic as an alternative solving method for the relaxed problem.
2. We propose a scaling technique with an arbitrary divisor scaling in order to reduce the computational complexity of the relaxed problem. We discuss its theoretical properties and compare it with GCD (greatest common divisor) scaling.
3. We propose an alternative aggregation-based relaxation method with adjustable computation complexity, discuss its properties and provide an acceleration technique to reduce the computation time.
4. We test and illustrate the numerical properties of the proposed methods through computer experiments.

The remainder of this paper is structured as follows. Section 2 contains the formulation of the primal problem. Section 3 offers a brief literature overview. Section 4 describes the basic Lagrangian relaxation for the problem considered and its decomposition. Section 5 discusses an alternative method for solving the upper problem. Section 6 proposes two scaling techniques to reduce the computational complexity of the problem. Similarly, Section 7 offers an alternative Lagrangian relaxation and discusses its properties and implementation. Section 8 contains the results of computer experiments. Finally, Section 9 presents the conclusions.

## 2. Problem

The problem of minimizing the total weighted tardiness on a single machine is a classical scheduling problem and is known to be strongly NP-hard (Lenstra *et al.*, 1977). The problem is denoted as $1||\sum w_iT_i$ in Graham's notation (Graham *et al.*, 1979) and is formulated as follows. We are given a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of $n$ jobs, with processing time $p_i > 0$, deadline $d_i > 0$ and weight $w_i > 0$ given for each job $i \in \mathcal{N}$, to process on the single machine. The goal is to determine the sequence of job starting times $S = (S_1, S_2, \ldots, S_n)$ that minimizes the total weighted tardiness of all jobs:

$$\min_S \sum_{i=1}^{n} w_i T_i, \tag{1}$$

where $T_i \stackrel{\text{def}}{=} [S_i + p_i - d_i]^+$, $[x]^+ \stackrel{\text{def}}{=} \max\{x, 0\}$ is the tardiness of job $i$.

Despite $T_i$ being a function of $S_i$, we will henceforth refer to $T_i$ instead of $[S_i + p_i - d_i]^+$. Furthermore, a machine can only process at most one job at any given moment. Thus our solution $S$ has to meet the following constraint:

$$\forall_{i,j\in\mathcal{N},\, j\neq i}: \quad (S_i + p_i \leq S_j) \vee (S_j + p_j \leq S_i). \tag{2}$$

Due to the regularity of the objective function for such a problem, a schedule $S$ is always left-shifted on the time axis. Thus, $S$ can be unambiguously represented by a processing order (permutation) of jobs $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, where $\pi(i)$ is the job that will be processed as the $i$-th. Thus, for a given processing order $\pi$ the schedule $S$ is computed recursively in time $O(n)$ as follows:

$$S_{\pi(1)} = 0, \tag{3}$$

$$S_{\pi(i)} = S_{\pi(i-1)} + p_{\pi(i-1)}, \qquad i = 2, 3, \ldots, n. \tag{4}$$

A more complete survey on the problem of minimizing the total tardiness on a single machine and its extensions in presented by Koulamas (2010).

## 3. Literature overview

The use of Lagrangian relaxation (LR) for optimization problems can be tracked back to 1963, when it was first applied to the resource allocation problem (Everett, 1963). Since then, due to its generality, the concept has been applied to a wide range of optimization problems. In this section we will focus on several such applications of LR, including both recent and older approaches.

We will start with papers related to scheduling. Fisher (1976) used LR for tardiness minimization for a single-machine scheduling problem. The results applied to a branch and bound (B&B) algorithm showed that the approach yields sharp lower bounds. A similar approach was employed by Potts and Van Wassenhove (1985). Instead of using a subgradient method, the authors chose a multiplier adjustment method, leading to fast calculation of the bounds. Next, LR was used to solve a hybrid flow-shop problem to minimize the total tardiness by Nishi *et al.* (2010). The authors relaxed the machine capacity constraint, decomposed the problem and then used dynamic programming (DP) to solve the resulting subproblems. A similar approach for the job-shop scheduling problem was considered by Chen and Luh (2003). However, the authors chose to relax the operation precedence constraint instead, resulting in fewer Lagrangian multipliers and reduced computational complexity.

In another paper, LR was applied to a stochastic scheduling problem in a hydrothermal system under uncertainty (Nowak and Römisch, 2000). The authors decomposed the problem into hydro and thermal subproblems and solved them using DP and a descent algorithm. The approach was evaluated on scenarios based on a real-life German power utility. In another paper, a heuristic algorithm based on LR and a subgradient method for joint train scheduling and the maintenance problem was presented (Zhang *et al.*, 2020). The authors applied the algorithm to a practical problem in the Chinese

railway network. A similar approach using a heuristic LR-based algorithm was shown for the joint problem of crane assignment and scheduling (Fu and Diabat, 2015). Another paper considered LR for scheduling for a steelmaking casting process (Cui and Luo, 2017). The authors showed the convergence proof under reasonable assumptions and proposed both a subgradient method and a simple heuristic.

More recently, LR was applied to scheduling for the problem of charging electric buses by Huang *et al.* (2023). The authors discretized decision variables to deal with nonlinearity of the changing process and performed a decomposition with respect to individual vehicles. Finally, LR was used for flexible job-shop scheduling with machine breakdowns and multiple optimization criteria ranging from tardiness to CO2 emission and noise pollution by Hajibabaei and Behnamian (2023). Following model linearization, the authors showed how LR helped reduce the problem complexity.

Aside from scheduling, LR is often used to solve routing problems, such as the vehicle routing problem (VRP). For example, LR was applied to the VRP in the context of pickup and delivery of containers in an inter-modal terminal. The resulting subgradient method was tested on a variety of problem instances. As another example, LR was applied to a VRP for minimizing gas emissions by Zhou and Lee (2017). The authors considered a realistic set of pollution-affecting factors and employed LR to reduce the complexity of the resulting formulation. Recently, LR approach was used for multi-compartment vehicle routing by relaxing task allocation constraints (Song *et al.*, 2024). Both DP and subgradient methods were used to solve the resulting dual problem. Regarding recent routing problems, LR was used for quality-of-service multicast routing in vehicular networks (Araújo *et al.*, 2024) as well as for a capacitated arc routing problem in the state-space-time network (Song *et al.*, 2023).

Aside from scheduling and routing, LR was applied to many other optimization problems. Examples include general integer programming (Fisher, 2004), resource allocation (Gocgun and Ghate, 2012), min-flow problems (Butt and Collins, 2013), classification (Gaudioso *et al.*, 2017) and supply chains (Ali *et al.*, 2023), among others.

Finally, we also provide a brief overview of classical (non-LR) approaches to the $1||\sum w_iT_i$ problem. Although rare, exact approaches to this problem are still considered, as shown by Speckenmeyer *et al.* (2023). The authors proposed a parallel branch-and-price method and showed its performance through experiments on benchmarks.

An example of a local search metaheuristic applied to $1||\sum w_iT_i$ was shown by Bilge *et al.* (2007). The authors proposed a deterministic tabu search method with a hybrid neighborhood and investigated the influence of several variants on the effectiveness of the algorithm. An effective simulated annealing approach was also proposed (Matsuo *et al.*, 1989). In another paper, an unusual exponential size local-search neighborhood was considered (Congram *et al.*, 2002). The authors showed how to search this exponential-size neighborhood in polynomial time and applied an iterated variant of the algorithm. For additional local-search approaches to $1||\sum w_iT_i$ please refer to Crauwels *et al.* (1998).

Aside from the above, other heuristics, ranging from priority rules to more sophisticated approaches, were proposed for this problem as shown, for example, by Potts and Van Wassenhove (1991). Interestingly, for a non-weighted variant of the $1||\sum w_iT_i$ problem, pseudopolynomial algorithms were proposed, as shown, for example by Lawler (1977).

Regarding more recent approaches, a hybrid evolutionary approach was proposed by Zakharova (2023). The author showed that the optimized operators used allowed obtaining results with quality matching existing works. A previously unconsidered variant of the total weighted tardiness problem with preemption was considered in another paper (Simanchev and Urazova, 2023). Finally, several variants of the problem with unknown complexity were proven to be pseudopolynomial (Zhao and Yuan, 2023).

## 4. Lagrangian relaxation

Lagrangian relaxation is based on relaxing some of the constraints of the primal problem and including them into the objective function as a penalty instead. In our case we relax the constraint (2), allowing the machine to process an arbitrary number of jobs at the same time. From now on we also assume that processing times $p_i$ are integers. Due to the fact that $p_i > 0$ and the formula (4), this immediately implies that

$$p_i \in \mathbb{N}_+, \tag{5}$$

$$S_i \in \mathbb{N}_0, \tag{6}$$

where $\mathbb{N}_0$ and $\mathbb{N}_+$ are sets of natural numbers with and without zero, respectively. Since job completion times are, by definition, $C_i = S_i + p_i$, this further implies that

$$C_i \in \mathbb{N}_+. \tag{7}$$

As a result, job starting and completion times are integers from the set $\{0, 1, \ldots, H\}$, where $H$ is the upper bound on the scheduling horizon. In practice, $H = \sum_{i=1}^{n} p_i$. Those assumptions impose a discrete character of time $t = 0, 1, 2, \ldots, H$. Thus we can narrow our analysis of the violation of machine capacity to unit-length intervals in the form of $(t-1, t]$ for $t = 1, 2, \ldots, H$.

Let $S$ be a schedule and $(t-1, t]$ be the interval of time for some $t$. We define the set of jobs which are

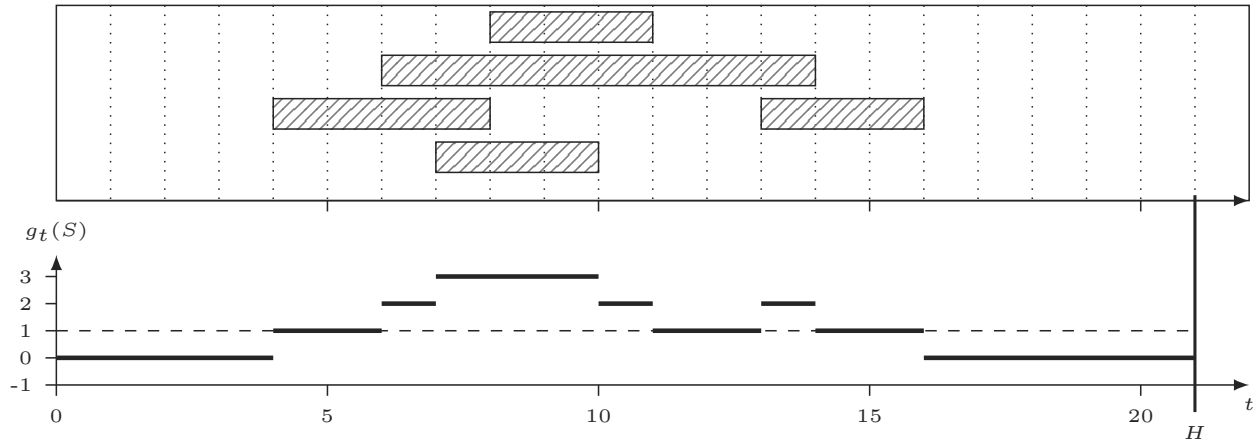


Fig. 1. Illustrative relaxed schedule and the corresponding function $g_t(S)$.

processed in $(t-1,t]$ as the ones satisfying simultaneously conditions $S_i \leq t-1$ and $C_i \geq t$. Since $S_i < t$ and $C_i = S_i + p_i$, by combining these two inequalities we introduce the set of jobs processed in the interval $(t-1,t]$;

$$\mathcal{I}_t(S) \stackrel{\text{def}}{=} \{i \in \mathcal{N} : t > S_i \geq t - p_i\}, \tag{8}$$

and the number of jobs processed in that interval:

$$g_t(S) \stackrel{\text{def}}{=} |\mathcal{I}_t(S)|. \tag{9}$$

An interpretation of function $g_t(S)$ for $t = 1, \ldots H$ is shown in Fig. 1. For the relaxed schedule $S$ presented in the top panel of the figure, functions $g_t(S)$ for successive instances of $t$ are shown in the bottom panel of the figure.

The problem from Section 2 can now be re-stated as a nonlinear optimization case:

$$\min_S \sum_{i=1}^{n} w_i T_i \tag{10}$$

subject to the constraints

$$g_t(S) = 1, \qquad t = 1, 2, \ldots, H, \tag{11}$$

$$0 \leq S_i \leq H - p_i, \qquad i = 1, 2, \ldots, n. \tag{12}$$

The constraint (11) ensures that exactly one job is processed in time interval $(t-1,t]$. The constraint (12) can be formulated by introducing the following set:

$$\mathbb{S} \stackrel{\text{def}}{=} \{S = (S_1, S_2, \ldots, S_n) : 0 \leq S_i \leq H - p_i, \;\; i = 1, 2, \ldots, n\}. \tag{13}$$

We will now perform the actual relaxation by replacing the constraint (11) with a penalty in the objective function in the form of dual variables $u \stackrel{\text{def}}{=} (u_1, u_2, \ldots, u_H) \in \mathbb{R}^H$, where $u_t$ is assigned for a fixed $t$. $\mathbb{R}^H$ is the real coordinate space of dimension $H$. We can perceive $u_t$ as the cost of using the machine in time interval $(t-1,t]$. As a result, we obtain the following Lagrangian function:

$$L(S,u) \stackrel{\text{def}}{=} \sum_{i=1}^{n} w_i T_i + \sum_{t=1}^{H} u_t (g_t(S) - 1). \tag{14}$$

The relaxed problem is formally stated as follows:

$$W(u) \stackrel{\text{def}}{=} \min_S L(S,u) \tag{15}$$

subject to the constraints

$$0 \leq S_i \leq H - p_i, \quad i = 1, 2, \ldots, n. \tag{16}$$

Next, we transform (14) into a more convenient form:

$$L(S,u) = \sum_{i=1}^{n} \Big( w_i T_i + \sum_{t=S_i+1}^{S_i+p_i} u_t \Big) - \sum_{t=1}^{H} u_t \stackrel{\text{def}}{=} \sum_{i=1}^{n} L_i(S_i,u) - U_H, \tag{17}$$

where

$$L_i(S_i,u) \stackrel{\text{def}}{=} w_i T_i + \sum_{t=S_i+1}^{S_i+p_i} u_t = w_i T_i + U_{S_i+p_i} - U_{S_i} \tag{18}$$

and

$$U_t \stackrel{\text{def}}{=} \sum_{s=1}^{t} u_s, \qquad t = 1, 2, \ldots, H. \tag{19}$$

**Remark 1.** In the problem (10)–(12), for each left-shifted *feasible* schedule $S$ and all $t = 1, 2, \ldots, H$, there must be $g_t(S) = 1$. In such a case, the expression (14) is equal to the objective function $\sum_{i=1}^{n} w_i T_i$ for any $u$, namely,

$$\sum_{i=1}^{n} w_i T_i = \sum_{i=1}^{n} w_i T_i + \sum_{t=1}^{H} u_t (g_t(S) - 1). \tag{20}$$

Using this remark, we can show that the solution of the relaxed problem (15)–(16) is a lower bound (LB) for the solution of the primal problem (10)–(12), i.e., for all $u$ we have

$$\sum_{i=1}^{n} w_i T_i^* \geq \min_{S\in\mathbb{S}} L(S,u), \tag{21}$$

where $T^* = (T_1^*, T_2^*, \ldots, T_n^*)$ is the tardiness vector calculated from the optimal solution $S^*$.

Inequality (21) is true, because $S^*$ can be applied to the relaxed problem as $S^* \in \mathbb{S}$. Moreover, $S^*$ is feasible. Therefore, based on Remark 1, we get

$$\begin{aligned}\sum_{i=1}^{n} w_i T_i^* &= \sum_{i=1}^{n} w_i T_i^* + \sum_{t=1}^{H} u_t(g_t(S^*)-1)\\ &\stackrel{\text{def}}{=} L(S^*,u).\end{aligned} \tag{22}$$

Thus $\min_{S\in\mathbb{S}} L(S,u)$ is less than or equal to $L(S^*,u)$.

Since we are interested in the highest possible LB, we would like to maximize $W(u)$:

$$\max_{u\in X\subseteq R^H} W(u) \leq \max_{u\in R^H} W(u) \leq L(S^*,u) \tag{23}$$

either over $u \in \mathbb{R}^H$ or over $u \in X$, where $X \subseteq \mathbb{R}^H$. The latter case is more attractive since it allows potential application of metaheuristics.

In general, the above model can be used to solve the primal problem (10)–(12) due to the duality principle. Let $S^*$ be the optimal solution to the relaxed problem for some $u$, i.e., $\min_S L(S,u) = L(S^*,u)$. If $g_t(S^*) = 1$, $t = 1,2,\ldots,H$, then $S^*$ is the optimal solution to the primal problem. Secondly, even if the resulting schedule $S$ is not optimal, it can be used to construct (i.e., heuristically) a feasible schedule that might provide a satisfactory solution to the primal problem. Finally, $W(u)$ itself can be used to provide an LB to evaluate the quality of solutions provided by other methods, e.g., beam search or as a lower bound employed in the potential algorithm of the B&B type. Because in recent years B&B has been treated as an ineffective optimization technology, we do not develop this research direction further.

While the problem (23) can be considered as-is, it is more convenient to decompose it into two subproblems which can be tackled separately: $\max_{u\in X\subseteq\mathbb{R}^H} W(u)$, henceforth called the upper (level) problem, and $\min_S L(S,u)$, henceforth called the lower (level) problem.

The upper problem is a continuous optimization problem with a nonlinear (piecewise linear, in fact) weakly convex function that is not differentiable at "interval junction" points. Moreover, the upper problem has a large size due to the fact that the vector $u$ has $H$ elements, which is usually much larger than $n$. Formally, it can be reformulated as linear programming, but the resulting size makes such a reformulation too complex and impractical. For example, for $H = 1000$ and $n = 100$, the expected formulation has $10^3$ variables and $10^{300}$ constraints. Due to function $W(u)$ being nondifferentiable, we will discuss several alternative approaches in the sequel to solve the upper problem.

As for the lower problem, due to (17), function $L(S,u)$ is separable with respect to $S$. Each sum component in $L(S,u)$ depends on single $S_i$ and the components are independent. Thus, minimization can be decomposed. Therefore, by applying (13)–(14) and (17)–(19) for fixed $u$, we obtain

$$\begin{aligned}\min_{S\in\mathbb{S}} L(S,u) &= \min_{S\in\mathbb{S}} \sum_{i=1}^{n} L_i(S_i,u) - U_H\\ &= \sum_{i=1}^{n} \min_{0\leq S_i\leq H-p_i} L_i(S_i,u) - U_H\\ &= \sum_{i=1}^{n} V_i(u) - U_H,\end{aligned} \tag{24}$$

where

$$\begin{aligned}V_i(u) &\stackrel{\text{def}}{=} \min_{0\leq S_i\leq H-p_i} L_i(S_i,u)\\ &= \min_{S_i=0,1,\ldots,H-p_i} L_i(S_i,u).\end{aligned} \tag{25}$$

For fixed $u$, each problem (25) can be solved by directly evaluating all possible starting times $S_i$. Since the formula (19) can be stated in a recursive form $U_t = U_{t-1} + u_t$, $t = 1,2,\ldots,H$, $U_0 = 0$, evaluation of $U_t$ can be done in time $O(H)$. Thus, determining (18) with the use of (25) requires time $O(H)$, while determining $W(u)$ through the formula (24) for fixed $u$ requires time $O(nH)$. The lower problem is usually solved multiple times for different values of $u$. It is important to note that, unlike the upper problem, the lower problem has to be solved optimally to ensure that the resulting value is a valid LB, as per (21). Due to the decomposition, the inexact approach to the upper problem, if applied, will not affect the requirements of the lower one.

## 5. Upper problem

We will now discuss several possible solving methods for the upper problem $\max_{u\in X\subseteq\mathbb{R}^H} W(u)$ for various $X$'s.

**5.1. Subgradient method.** The subgradient (SUB) method is often applied to Lagrangian relaxation problems (Held *et al.*, 1974). It is recommended chiefly to find $\max_u W(u)$, where $W(u)$ is a nondifferentiable function with a limited scope of singularities. Briefly speaking, SUB can be perceived as an iterative method that produces a sequence of vectors $u^0, u^1, u^2, \ldots$ through

the following formula:

$$u_t^{k+1} = u_t^k + \alpha^k(g_t(S^k) - 1), \quad t = 1, 2, \ldots, H, \quad (26)$$

where $\alpha^k$ is the step length in the $k$-th iteration while $S^k$ is an optimal solution to the lower problem in the $k$-th iteration, i.e., $\min_S L(S, u^k) = L(S^k, u^k)$. The choice of sequence $\alpha^k$ heavily affects the stability, convergence and convergence speed of the method.

In theory, the convergence of LB to the value of the UB is guaranteed for any sequence such that $\lim_{k\to\infty} \alpha^k = 0$, $\sum_{k=1}^{\infty} \alpha^k = \infty$ (e.g., harmonic sequence $\alpha^k = c/k$ for constant $c$). The choice of $c \stackrel{\text{def}}{=} \alpha^1$ remains an open question, as this value depends heavily on specific problem instances and their size. The method thus requires tedious parameter tuning. The lack of growth of $W(u^k)$ for $k = 1, 2, \ldots$ might be caused by the "zig-zagging" phenomenon accompanied by $\alpha^k$ quickly approaching zero. Similar results were obtained for a slower converging sequence $\alpha^k = c/k^\gamma$, $\gamma \ll 1$. Another convergence-guaranteed sequence is

$$\alpha^k = \gamma^k \frac{W^* - W(u^k)}{||g(S^k) - 1||^2}, \quad (27)$$

where $W^*$ is an optimal value for the objective function. The denominator contains the norm of constraint violation for $S^k$, for example, an Euclidian norm:

$$||g(S^k) - 1||^2 = \sum_{t=1}^{H} (g_t(S^k) - 1)^2. \quad (28)$$

For $\alpha^k$ given by the formula (27), the convergence was proven. In practice, due to $W^*$ being unknown, some upper bound $\text{UB}^k$ is used, even though it does not guarantee convergence. It is advised to update $\text{UB}^k$ in each iteration $k$ by using an *auxiliary* heuristic. One of the simplest approaches it computing the value of the objective function based on (3) and (4) for a permutation $\pi$ obtained by ordering all jobs according to a nondecreasing value of $S_i^k$. It is possible to use more complex methods (i.e., metaheuristics) instead, resulting in a better approximation of $\text{UB}^k$ but a slower total algorithm running time. The topic of convergence for variants of the dual method remains an active field of research (Bragin *et al.*, 2015).

Due to the inequality (21), the best lower bound after performing $k$ first SUB iterations equals $\text{LB}^k = \max\{W(u^0), W(u^1), \ldots, W(u^k)\}$. Similarly, the best upper bound is $\text{UB} = \min\{\text{UB}^1, \text{UB}^2, \ldots, \text{UB}^k\}$.

**5.2. Local-search metaheuristic.** The aim is to find $\max_u W(u)$ using a metaheuristic. Generally speaking, any metaheuristic can be used, and no assumptions about function $W(u)$ are important. In this paper, we propose a method created by modifying the well-known simulated annealing (SA) local-search metaheuristic. The resulting method will be called modified SA, or m-SA for short.

The method is iterative, producing a vector sequence $u^0, u^1, \ldots$ in much the same way as in the conventional SA, but with a different method of generating the next random solution. Vector $u^k$ is our current solution, with $W(u^k) = \min_S L(S, u^k) = L(S^k, u^k)$ being its objective function. By $N(u^k)$ we denote the neighborhood of $u^k$ defined by us as follows:

$$N(u^k) = \{u = (u_1, \ldots u_H) : u_t \in [u_t^k - A, u_t^k + (n-1)A], \; t = 1, \ldots, H\}, \quad (29)$$

where $A$ is some number called elementary penalty growth and the domain for values $u_t$ is defined by continuous intervals. The next solution, $u^{k+1} \in N(u^k)$, is selected from the neighborhood of the current solution $u^k$ as follows. First, we randomly generate a single perturbed solution $\tilde{u}$

$$\tilde{u}_t = u_t^k + Z_t \cdot (g_t(S^k) - 1), \quad t = 1, 2, \ldots, H, \quad (30)$$

where $Z_t$, $t = 1, \ldots, H$ are random numbers with the uniform distribution on the interval $[0, A]$. This perturbed solution choice is unlike the typical SA, as the neighborhood is not uniform but also dynamic due to the term $g_t(S^k)$. Next, we proceed as in the regular SA: we compute $\Delta = W(\tilde{u}) - W(u^k)$. If $\Delta \leq 0$, then solution $\tilde{u}$ is accepted, i.e., $u^{k+1} := \tilde{u}$. Otherwise, solution $\tilde{u}$ is accepted with probability $e^{-\Delta/T}$, where $T$ is the parameter called temperature. If solution $\tilde{u}$ has not been accepted by either the first or second condition, then we set $u^{k+1} := u^k$.

The temperature in SA is reduced systematically according to the so-called cooling scheme, usually by iterations. In our case, we employ a geometric cooling scheme $T^{k+1} = \lambda T^k$, $k = 0, 1, \ldots$, with a constant $\lambda < 1$. Too small $\lambda$ causes premature convergence to a local extreme of poor quality, too large one implies long computational time. The resulting m-SA algorithm has several tuning parameters, such as $u^0$, $T^0$, $\lambda$ or the stop condition, which have to be set experimentally.

Since the values of $W(u^k)$ change "randomly" for subsequent $k$, the final answer of m-SA is determined as $\text{LB} = \max\{W(u^0), W(u^1), \ldots\}$. Additionally, in each iteration $k$, we can employ an auxiliary heuristic to order jobs according to nondecreasing values $S_i^k$ (just as with the SUB method). In this way, we can obtain a sequence of UBs and compute $\text{UB} = \min\{\text{UB}^0, \text{UB}^1, \ldots, \text{UB}^k\}$.

**5.3. Hybrid approaches.** Here we will discuss a few modifications of SUB and m-SA in order to improve their computation complexity and convergence. We will

start with modifications to the model (27) which can include: (i) initializing $u^0$ with random values, (ii) guided control of sequence $\gamma^k$, (iii) random control of sequence $\gamma^k$. Option (i) is obvious, as zero values in $u$ result in unfavorable behavior of $W(u)$ in the initial phase for both SUB and m-SA. This is caused by $\min L(S,U)$ overlapping all jobs at time 0, which results in $W(u)$ failing to increase at first. We will now consider options (ii) and (iii) to increase the speed of convergence of $W(u)$ to the UB.

Guided control is aimed at automatically adjusting values $\gamma^k$ to a given instance to shorten the initial phase with unfavorable values $W(u)$. This can be achieved in various ways. One often employed method is to set $\gamma^0 \approx 2.0$ and then reduce $\gamma^k$ by some factor if there was no improvement of $W(u^k)$ in the last 5 iterations. Another method uses a lower-raise principle as follows. If $W(u^k)$ decreases, then we set $\gamma^{k+1} = \sigma \cdot \gamma^k$, $\sigma < 1$; if it increases, then we set $\gamma^{k+1} = \tau \cdot \gamma^k$, $\tau > 1$. Values $\sigma$ and $\tau$ should be close to 1.

Random changes of $\gamma^k$ can be seen as another variant of SA or a hybrid between variable neighborhood search and random search. The approach is based on introducing a small random perturbation to $\gamma^k$ in the formula (27). The resulting random variable $\gamma^k$ has the average value change complying with (27). Practice shows that such randomness prevents the "zig-zag" pattern specific to SUB from occurring.

## 6. Scaling

Computational complexity $O(nH)$ of the lower problem $W(u) = \min_S L(S,u)$ is considered unacceptably high if $H$ is large. In this section we discuss some time-scaling approaches, meant to reduce the computational complexity of this problem.

**6.1. GCD scaling.** We start by discussing GCD (greatest common divisor) scaling. It should be noted, however, that this approach is not meant as a viable practical technique. We introduce it only as a foundation and reference for the arbitrary scaling that will be presented in the next subsection.

Using the GCD notion, we define

$$a \stackrel{\text{def}}{=} \text{GCD}(p_1, \ldots, p_n, d_1, \ldots, d_n). \tag{31}$$

Then, we introduce the new problem with the reduced size in the scale $1 : a$ by the one-to-one transformation given below:

$$p'_i = \frac{p_i}{a}, \quad d'_i = \frac{d_i}{a}. \tag{32}$$

Note that $p'_i$ and $d'_i$ are integers. By the assumption (32) and the fundamentals of scaling, we conclude that

$$S'_i = \frac{S_i}{a}, \quad H' = \sum_{i=1}^{n} p'_i = \frac{H}{a}. \tag{33}$$

For this reduced problem, we mark the appropriate values with prime, i.e., $u'$, $L'(S',u')$, and $W'(u')$. The scaled problem analogous to (23) can now be written as

$$\max_{u'} W'(u') = \max_{u'} \min_{S'} L'(S',u'). \tag{34}$$

One can ask about the relation between $L'(S',u')$ and $L(S,u)$ from (17). First, note that finding $\min_{S'} L'(S',u')$ for a given $u'$ requires $O(nH/a)$ time, compared with $O(nH)$ for the previously defined $L(S,u)$ in (17). Next, using a known property for $[\frac{x}{c}]^+ = \frac{1}{c}[x]^+$ for $c > 0$ and then applying (32), we can obtain

$$L'(S',u') = \frac{1}{a} L(S,\hat{u}), \tag{35}$$

where $\hat{u}$ is an *extension* of $u'$ defined as follows:

$$\hat{u_t} = u'_r, \quad t = (r-1)a + 1, \ldots, ra, \tag{36}$$

$r = 1, \ldots H'$. Note that $\hat{u}_t$ is constant in intervals of length $a$. The correctness of (35) is shown in Appendix.

By analogy to (17)–(19), the formula (35) can be calculated in a faster way, which speeds up the process of finding $\min_{S'} L'(S',u')$:

$$\begin{aligned} L'(S',u') &= \sum_{i=1}^{n} (w_i T'_i + \sum_{r=S'_i+1}^{S'_i+p'_i} u'_r) - \sum_{r=1}^{H'} u'_r \\ &= \sum_{i=1}^{n} L'_i(S'_i,u') - U'_{H'}, \end{aligned} \tag{37}$$

where

$$\begin{aligned} L'_i(S'_i,u') &\stackrel{\text{def}}{=} w_i T'_i + \sum_{t=S'_i+1}^{S'_i+p'_i} u'_t \\ &= w_i T'_i + U'_{S'_i+p'_i} - U'_{S'_i} \end{aligned} \tag{38}$$

and

$$T'_i = [S'_i + p'_i - d'_i]^+, \quad i = 1,2,\ldots,n, \tag{39}$$

$$U'_r = \sum_{s=1}^{r} u'_s, \quad r = 1,2,\ldots,H'. \tag{40}$$

In practice, this means that the calculations are performed completely on the scaled data, namely, these with prime superscripts.

Next, we will set the relation between $W(u')$ and $W(u)$ from (21). From (35), it follows that

$$W'(u') = \frac{1}{a} W(\hat{u}). \tag{41}$$

Lastly, we are interested in the relation between $\max_{u'} W(u')$ and $\max_u W(u)$ from (23). From the relation between $\hat{u}$ and $u$, we obtain

$$\max_{u'} W(u') = \frac{1}{a} \max_{\hat{u}} W(\hat{u}) \leq \frac{1}{a} \max_{u} W(u). \quad (42)$$

One can hope that $a \cdot \max_{u'} W(u')$ approximates $\max_u W(u)$ with sufficiently high accuracy under slightly smaller computational complexity. The drawback of this scaling technique is the risk that $a = 1$ for the majority of instances. Also, due to the inequality (42), GCD scaling should be considered a relaxation.

**6.2. Arbitrary scaling.** In this section we propose alternative relaxation to avoid the pessimistic case of GCD $= 1$. In order to distinguish cases considered in the paper, we mark this relaxed/scaled problem with the double prime. Let $b$ be an integer. Having in mind the definition of relaxation, we introduce the scaled data as follows:

$$p_i'' = \left\lfloor \frac{p_i}{b} \right\rfloor, \quad d_i'' = \left\lceil \frac{d_i}{b} \right\rceil. \quad (43)$$

Operators $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ mean ceil and floor roundings, respectively. Note that $p_i'' b$, $d_i'' b$, $i = 1, \ldots, n$ have GCD $= b$, so the results from the previous subsection can be applied in full; however, this simultaneously introduces an additional relaxation. There still remains the question of relaxation depth. As the consequence of (43), we have

$$S_i'' = \left\lfloor \frac{S_i}{b} \right\rfloor, \quad H'' = \sum_{i=1}^{n} p_i'' \leq \frac{H}{b}. \quad (44)$$

Let us analyze the relation between scaled and non-scaled problems. By analogy to (17), we obtain

$$L''(S'', u'') = \sum_{i=1}^{n} \Big( w_i T_i'' + \sum_{r=S_i''+1}^{S_i''+p_i''} u_r'' \Big) - \sum_{r=1}^{H''} u_r''. \quad (45)$$

Using obvious inequalities $\lfloor x \rfloor \leq x$ and $\lceil x \rceil \geq x$, we have $T_i'' \leq T_i/b$, which establishes the relation between the first term in (45) and (21). In order to show successive dependencies, we refer to the *extension* of $u''$ to $\tilde{u}$ defined as follows:

$$\tilde{u}_t = u_r'', \quad t = (r-1)\, b, \ldots, r\, b, \quad r = 1, \ldots H''. \quad (46)$$

From (46), we can deduce that

$$\sum_{t=(r-1)b+1}^{rb} \tilde{u}_t = b u_r'', \quad r = 1, 2, \ldots, H''. \quad (47)$$

Let us analyze the formal relation of components appearing in (45). Using (47), we can find that

$$\sum_{r=1}^{H''} u_r'' = \frac{1}{b} \sum_{r=1}^{H''} b u_r'' = \frac{1}{b} \sum_{r=1}^{H''} \sum_{t=(r-1)b+1}^{rb} b \tilde{u}_t$$

$$= \frac{1}{b} \Big( \sum_{t=1}^{b} \tilde{u}_t + \sum_{t=b+1}^{2b} \tilde{u}_t + \sum_{t=2b+1}^{3b} \tilde{u}_t + \ldots + \sum_{t=(H''-1)b+1}^{bH''} \tilde{u}_t \Big) = \frac{1}{b} \sum_{t=1}^{bH''} \tilde{u}_t. \quad (48)$$

Since $bH'' \leq H$, the component (48) formally has to be modified as follows:

$$\sum_{r=1}^{H''} u_r'' = \frac{1}{b} \sum_{r=1}^{H''} b u_r'' = \frac{1}{b} \sum_{t=1}^{bH''} \tilde{u}_t = \frac{1}{b} \sum_{t=1}^{H} \tilde{u}_t - \frac{1}{b} \sum_{t=bH''+1}^{H} \tilde{u}_t. \quad (49)$$

The first equality in (49) is a technical adjustment, the second employs the definition of $\tilde{u}$, and the third decomposes the sum in index intervals $[0, bH'']$ and $[bH'', H]$, such that $[bH'', H] \subseteq [0, H]$. Due to the fact that $bH'' \approx H$, the correction provided in the last term in (49) can be considered relatively small; however, an error potentially exists. This ends the evaluation of the last component in (45). To evaluate the penultimate component in (45), we deduce that from (47):

$$\sum_{r=S_i''+1}^{S_i''+p_i''} u_r'' = \frac{1}{b} \Big( b u_r''|_{r=S_i''+1} + b u_r''|_{r=S_i''+2} + b u_r''|_{r=S_i''+3} + \ldots + b u_r''|_{r=S_i''+p_i''} \Big)$$

$$= \frac{1}{b} \Big( \sum_{t=S_i''b+1}^{(S_i''+1)b} \tilde{u}_t + \sum_{t=(S_i''+1)b+1}^{(S_i''+2)b} \tilde{u}_t + \sum_{t=S_i''b+2}^{(S_i''+3b} \tilde{u}_t + \ldots + \sum_{t=(S_i''+p_i-1'')b+1}^{(S_i''+p_i'')b} \tilde{u}_t \Big)$$

$$= \frac{1}{b} \sum_{t=S_i''b+1}^{(S_i''+p_i'')b} \tilde{u}_t \approx \frac{1}{b} \sum_{t=S_i+1}^{S_i+p_i} \tilde{u}_t. \quad (50)$$

The last approximation is valid only if $S_i'' \approx S_i/b$ and $p_i'' \approx p_i/b$. Since minimization $\min_S'' L''(S'', u'')$ does not depend on $\sum_{r=1}^{H''} u_r''$, we have

$$W(u'') \leq \frac{1}{b} W(\tilde{u}), \tag{51}$$

$$\max_{u''} W(u'') \leq \frac{1}{b} \max_{\tilde{u}} W(\tilde{u}) \leq \frac{1}{b} \max_{u} W(u). \tag{52}$$

The computational complexity of the lower problem is $O(nH/b)$ and it depends on $b$. Assuming that $b = H/m$ for some $m$, we can obtain computational complexity $O(nm)$, which does not depend on $H$. It is attractive from the theoretical point of view; however, the approach suffers from the relaxation of the original problem. The quality of approximation $\max_u W(u)$ by $b \cdot \max_{u''} W(u'')$ depends on $b$ and should be tested experimentally. Actually, the solution to the problem $\min_{S''} L(S'', u'')$ is the lower bound to $\min_S L(S, u)$ with a small error, which has been already shown. Therefore, we propose the following procedure: (i) find $\max_{u''} \min_{S''} L''(S'', u'')$, using SUB with the UB calculated for the relaxed data problem, (ii) find the solution to the problem $\min_S L(S, u)$ for $u$ calculated in the previous step and the UB for the original data set. There still remains an open problem, how to find $\min_{S''} L(S'', u'')$ efficiently to avoid computational complexity $O(nH)$. This subject will be discussed in detail below.

## 7. Aggregation of constraints

In this section we provide an alternative form of relaxation through aggregation, leading to reduced computational complexity of the lower level problem.

**7.1. Formulation.** Let us define a sequence of integer time moments: $0 = t_0 < t_1 < t_2 < \ldots t_{m-1} < t_m = H$. Summing several conditions (11) as follows:

$$\sum_{t=t_{k-1}+1}^{t_k} g_t(S) = t_k - t_{k-1}, \quad k = 1, 2, \ldots, m, \tag{53}$$

we can introduce a new optimization problem,

$$\min_S \sum_{i=1}^{n} w_i T_i, \tag{54}$$

subject to the constraints

$$\frac{1}{t_k - t_{k-1}} \sum_{t=t_{k-1}+1}^{t_k} g_t(S) = 1, \quad k = 1, 2, \ldots, m, \tag{55}$$

$$0 \leq S_i \leq H - p_i, \quad i = 1, 2, \ldots, n. \tag{56}$$

Analyzing differences between (10)–(12) and (54)–(56), one can find that the condition (11) limits the number of jobs in each interval $(t-1, t)$, whereas the condition (55) limits the average length of jobs processed in the interval $(t_{k-1}, t_k)$. It is clear that (55) is a relaxation of (11).

Since the definition of expression $\sum_{t=t_{k-1}+1}^{t_k} g_t(S)$ does not offer any reduction of computational complexity, we will convert it to a more convenient form. To this end we define

$$h_k(S) \stackrel{\text{def}}{=} \frac{1}{t_k - t_{k-1}} \sum_{t=t_{k-1}+1}^{t_k} g_t(S), \quad k = 1, 2, \ldots, m. \tag{57}$$

Function $h_k(S)$ represents the average length of all jobs which are processed in the interval $(t_{k-1}, t_k)$. Note that this interval is composed of unit-time intervals of the form $(t_{k-1}, t_{k-1}+1)$, $(t_{k-1}+1, t_{k-1}+2)$, ..., $(t_k - 1, t_k)$, which in turn define constraints $g_{t_{k-1}+1}(S) = 1$, $g_{t_{k-1}+2}(S) = 1, \ldots, g_{t_k}(S) = 1$. We can calculate $h_k(S)$ by considering suitable pieces of jobs $i = 1, \ldots, n$ processed in $(S_i, S_i + p_i)$ and having a common part with the interval $(t_{k-1}, t_k)$. Thus, with the provided definition, we can introduce another expression on $h_k(S)$, namely,

$$h_k(S) = \sum_{i=1}^{n} h_{ki}(S_i), \quad k = 1, 2, \ldots, m, \tag{58}$$

where

$$h_{ki}(S_i) = \left[ \frac{\min\{S_i + p_i, t_k\} - \max\{S_i, t_{k-1}\}}{t_k - t_{k-1}} \right]^+ \tag{59}$$

and $[x]^+ = \max\{0, x\}$. The numerator in (59) represents the part of job $i$ processed in the interval $(t_{k-1}, t_k]$. From (59) we conclude that $0 \leq h_{ki}(S_i) \leq 1$. Using (57) we obtain the Lagrangian function with penalty $v_k$ for each constraint (55), $k = 1, 2, \ldots, m$,

$$M(S, v) = \sum_{i=1}^{n} w_i T_i + \sum_{k=1}^{m} v_k (h_k(S) - 1). \tag{60}$$

Substituting (58) to (60) and then changing the order of the summing operators, we finally obtain

$$\begin{aligned} M(S, v) &= \sum_{i=1}^{n} w_i T_i + \sum_{i=1}^{n} \sum_{k=1}^{m} v_k h_{ki}(S_i) - V \\ &= \sum_{i=1}^{n} [w_i T_i + \sum_{k=1}^{m} v_k h_{ki}(S_i)] - V \\ &\stackrel{\text{def}}{=} \sum_{i=1}^{n} M_i(S_i, v) - V, \end{aligned} \tag{61}$$

where

$$M_i(S_i, v) = w_i T_i + \sum_{k=1}^{m} v_k h_{ki}(S_i),$$
$$V = \sum_{k=1}^{m} v_k. \tag{62}$$

As before, the lower level problem provides a lower bound on the optimal solution and can be decomposed on separate subproblems:

$$\min_S M(S, v) = \sum_{i=1}^{n} \min_{S_i} M_i(S_i, v) - V. \tag{63}$$

For the proposed new relaxation, the formulas (26)–(28) for the upper level problem are no longer valid. The formulas below should be used instead:

$$v_k^{r+1} = v_k^r + \alpha^r (h_k(S^r) - 1), \tag{64}$$

$$\alpha^r = \gamma^r \frac{\mathrm{UB}^r - W(v^r)}{\sum_{k=1}^{m} (h_k(S^r) - 1)^2}, \tag{65}$$

where $S^r$ is the optimal solution of the lower level problem in the $r$-th iteration.

**7.2. Computation and complexity.** In this section we provide an efficient method of finding $\min_{S_i} M_i(S_i, v)$, from (62). We employ the fact that $M_i(S_i, v)$ is a piecewise linear function. In order to minimize a piecewise linear function $f(\cdot)$, it is sufficient to compute $\min_{b \in B} f(b)$, where $B$ is a set of breaking points of $f(\cdot)$. Moreover, if $f_1, f_2, \ldots, f_n$ are piecewise linear functions with breaking points defined by sets $B_1, B_2, \ldots, B_n$, respectively, then function $f = \sum_{i=1}^{n} f_i$ is also a piecewise linear function with breaking points in the set $B \subseteq \bigcup_{i=1}^{n} B_i$. Thus, to minimize $f$, it is sufficient to check all breaking points located in sets $B_1, \ldots, B_n$.

In consequence, to find $\min_{S_i} M_i(S_i, v)$, it is sufficient to check the breaking points of all its components. There are at most four breaking points for each component $v_k h_{ki}(S_i)$ and one for component $w_i T_i$; thus, we need to check $O(m)$ values of $S_i$. For each such $S_i$, the function $M_i(S_i, v)$ has to be computed. Its value can be computed in time $O(m)$ since $w_i$ and $v_k$ are constants, while $T_i$ and $h_{ki}$ can be computed in time $O(1)$. We conclude that $\min_{S_i} M_i(S_i, v)$ can be computed in time $O(m^2)$. The last elements from (63) that we need to account for are (i) component $V$, which can be computed in time $O(m)$, and (ii) the sum over $n$ jobs. This brings our computational complexity to $O(nm^2 + m) = O(nm^2)$.

**7.3. Acceleration.** The resulting complexity $O(nm^2)$ is smaller than $O(nH)$ only when $m < \sqrt{H}$. Now we will show an alternative computation method that allows us to compute $\sum_{k=1}^{m} v_k h_{ki}(S_i)$ in time $O(1)$ instead of $O(m)$.

The acceleration is based on the following idea. We know the time axis is divided into intervals $(0, t_1), (t_1, t_2), \ldots (t_{m-2}, t_{m-1}), (t_{m-1}, t_H)$. Let us consider job $i$ with starting time $S_i$ and completion time $S_i + p_i$. The job starts in some interval $x$ and ends in some interval $y$. This means that $t_{x-1} < S_i \leq t_x$ and $t_{y-1} < S_i + p_i \leq t_y$. Three cases are possible, as detailed below.

The first case is when $y = x$, i.e., job $i$ starts and ends in the same interval. In such a case, $h_{ki}(S_i) = 0$ for all $k \neq x$ and $\sum_{k=1}^{m} v_k h_{ki}(S_i)$ is reduced to a single term $v_x h_{xi}(S_i)$, which can be computed in $O(1)$.

The second case is when $y = x + 1$, i.e., job $i$ spans two subsequent intervals $x$ and $x + 1$. Then $h_{ki}(S_i) = 0$ for all $k \neq x$ and $k \neq y$, and $\sum_{k=1}^{m} v_k h_{ki}(S_i)$ is reduced to $v_x h_{xi}(S_i) + v_y h_{yi}(S_i)$ which is computed in time $O(1)$.

The last case is for $y > x + 1$, i.e., job $i$ spans for at least three intervals $x$ through $y$. As before, $h_{ki}(S_i) = 0$ for all $k < x$ or $k > y$ and $\sum_{k=1}^{m} v_k h_{ki}(S_i)$ is reduced to $\sum_{k=x}^{y} v_k h_{ki}(S_i)$. However, this takes time $O(y - x)$ to compute, which can be as large as $O(m)$. Let us notice, however, that all inner intervals $x + 1$ through $y - 1$ are always full, i.e., $h_{ki}(S_i) = 1$ for $k \in \{x+1, x+2, \ldots, y-1\}$. This means that

$$\sum_{k=x+1}^{y-1} v_k h_{ki}(S_i) = \sum_{k=x+1}^{y-1} v_k. \tag{66}$$

In order to compute (66) in time $O(1)$, we will use the $m \times m$ matrix $A = a_{[x,y]}$, where

$$a_{x,y} = \sum_{k=x}^{y} v_k. \tag{67}$$

In other words, $a_{x,y}$ contains the value of sum of $v_k$ for all intervals $x$ through $y$. Thus, our sum $\sum_{k=1}^{m} v_k h_{ki}(S_i)$ is reduced to $v_x h_{xi}(S_i) + a_{x+1,y-1} + v_y h_{yi}(S_i)$. Matrix $A$ contains prefix sums, i.e., $a_{x,y} = a_{x,y-1} + v_y$. Accordingly, $A$ can be constructed in time $O(m^2)$. As a result, the complexity of the algorithm was reduced from $O(nm^2)$ to $O(nm + m^2)$, and now offers a reduction compared to $O(nH)$ as long as $m < H$ and $m < \sqrt{nH}$. Further reductions might be possible.

It should also be noted that this algorithm requires us to be able to convert job starting and completion times $S_i$ and $S_i + p_i$ into interval numbers $x$ and $y$. To do it efficiently, we propose to construct vector $B$ such that $B_t$ is the interval number corresponding to time moment $t$, where $t \in \{0, 1, 2, \ldots, H-1, H\}$. With this, $x$ and $y$ can be determined in $O(1)$. The last issue is the construction

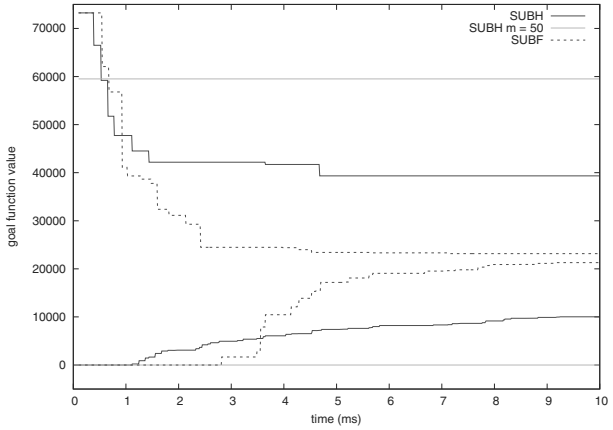


Fig. 2. Results for SUBH, SUBH $m = 50$ and SUBF using the illustrative instance.

of vector $B$. In theory, it takes time $O(H)$; however, $B$ does not change, so it can be created once on the upper level (e.g., in the subgradient method) and then passed to the lower problem as input no matter how many times the lower problem is invoked.

## 8. Computer experiments

Computer experiments have been performed in order to evaluate numerical properties of the proposed approaches. We considered two types of experiments: (i) illustrative experiments using a sample instance to highlight the dominant properties of our approaches, and (ii) large-scale tests performed on a number of instances.

All algorithms were coded in `C++` and compiled using `g++` 9.4.0 and `-O3` optimization flag. The source code of the project is accessible through the public GitHub repository (Idzikowski, 2023). The experiments were run on a machine with an Intel i7-11850H processor a with 2.5 GHz clock and 32 GB of RAM. All coding and experiments were carried out under Ubuntu 20.04 using the Linux 5.15 kernel.

With regards to problem instances, we have generated the instances using the scheme shown in OR-library (Beasley, 1990), which serves a source of benchmarks for a number of optimization problems, including the total weighted tardiness on a single machine. In order to portray GCD scaling, some instances were altered by multiplying all $p_i$ and $d_i$ values by 16. We will indicate for which instances this was done.

All figures in this section portray LB and UB trajectories obtained by the algorithms. In Figs. 2–7, time (in milliseconds) is represented on the horizontal axis and the goal function value on the vertical axis.

**8.1. Algorithms.** The following algorithms were tested in our experiments:

1. SUBH: the SUB method with sequence $\alpha^k = c/k$, $c = 1$.
2. SUBF: the SUB method with the formulas (27) and (28) applied, where $\gamma$ starts at 2 and is multiplied by $0.95$ if $W(u)$ has not improved in five iterations.
3. SUBR: the SUB method with formulas (27) and (28) applied, where $\gamma^k$ is a random variable from the uniform distribution on the interval $[0.95, 1.05]$.
4. m-SA: the m-SA method with $\lambda = 0.95$, $T^0 = 1000$.
5. DS: a descent method for the primal problem. This method is used as a comparison and represents a non-LR approach.

The additional specification in the form of $a = \ldots$ indicates the range of GCD scaling (e.g., SUBF $a = 4$). Similarly, $b = \ldots$ indicates the use of arbitrary scaling (e.g., SUBF $b = 5$). Finally, term $m = \ldots$ indicates the use of the aggregated relaxation from Section 7 (e.g., SUBF $m = 50$). For DS, the suffix indicates the neighborhood type:

1. $\mathrm{DS}_{\mathrm{swap}}$: neighborhood based on swapping (interchange) of job pairs.
2. $\mathrm{DS}_{\mathrm{adj}}$: neighborhood based on swapping (interchange) of adjacent job pairs.
3. $\mathrm{DS}_{\mathrm{ins}}$: neighborhood based on removing a job and inserting it into a different position in the permutation.

**8.2. Illustrative experiment.** In this experiment, we used one instance (generated according to the OR-library scheme as indicated earlier) with $n = 30$ and values $p_i$ and $d_i$ multiplied by 16 as described earlier. The illustrative instance is shown in Table 1. The optimal solution for this instance has the value of 22 864, using $a = 1$. By contrast, a standard greedy algorithm scheduling one job at a time yielded a value of 54 704 for this instance.

**SUB and SA.** Figures 2, 3 and 4 show trajectories for the SA method and various variants of the SUB approach. Due to a large number of variants, we split this into three separate plots, with SUBF shown in all three as a reference. Predictably, the SUBH method with the harmonic sequence performs badly despite its theoretical convergence guarantees. Surprisingly, the aggregated variant only compounds the result. The SUBF and SUBR methods provide much better results, and their aggregated variants have considerably higher convergence rates with little effect on the solution quality. Finally, the m-SA

Table 1. Illustrative instance.

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_i$ | 48 | 160 | 16 | 64 | 128 | 80 | 80 | 32 | 160 | 160 | 64 | 16 | 128 | 64 | 160 |
| $d_i$ | 496 | 1056 | 768 | 864 | 1072 | 1376 | 1120 | 1040 | 1072 | 1280 | 480 | 1008 | 1312 | 976 | 912 |
| $w_i$ | 10 | 4 | 5 | 2 | 5 | 9 | 4 | 10 | 10 | 9 | 4 | 3 | 5 | 10 | 1 |
| $i$ | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| $p_i$ | 96 | 32 | 160 | 80 | 80 | 16 | 128 | 80 | 32 | 64 | 32 | 128 | 48 | 32 | 48 |
| $d_i$ | 1376 | 592 | 656 | 1136 | 736 | 1056 | 1072 | 1376 | 480 | 640 | 560 | 1200 | 976 | 1392 | 592 |
| $w_i$ | 8 | 5 | 10 | 7 | 6 | 1 | 5 | 3 | 4 | 3 | 10 | 5 | 3 | 8 | 7 |

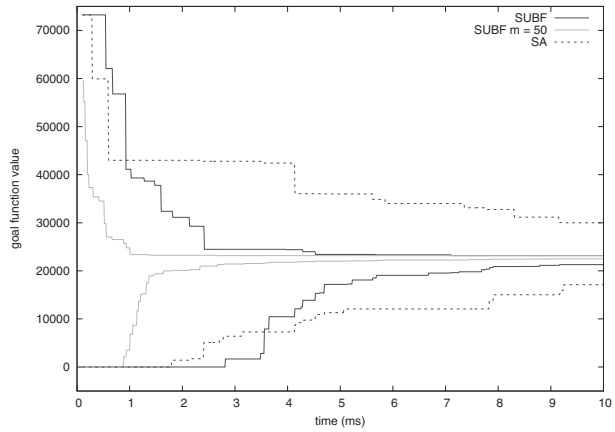


Fig. 3. Results for SUBF, SUBF $m = 50$ and m-SA using the illustrative instance.

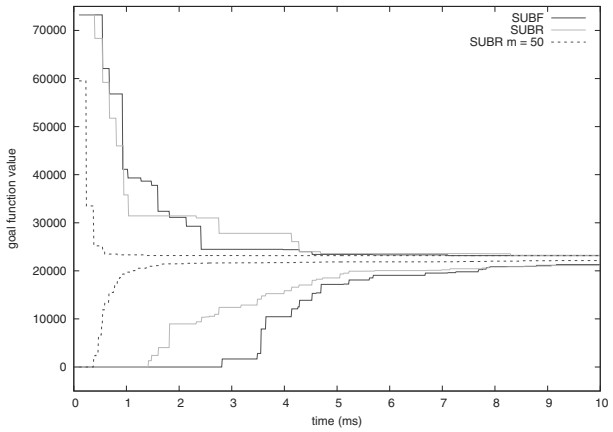


Fig. 4. Results for SUBF, SUBR and SUBR $m = 50$ using the illustrative instance.

method performs in-between SUBH and SUBF. It should be noted, however, that m-SA convergences faster in the early stage, making it more viable for use in time-sensitive situations.

**GCD scaling.** Figure 5 shows trajectories for possible GCD scaling for $a \in \{1, 2, 8, 16\}$ (all potential divisors). Regarding the algorithm running time, the observed acceleration was approximately $a$ times. Moreover, the observed trajectories in iterations were identical for all $a$. The main profit of this approach is to reduce the running time without the loss in LB and UB quality.

**Arbitrary scaling.** Figure 6 shows trajectories for arbitrary scaling for $b \in \{1, 5, 10, 15\}$ (we avoided too many cases to increase plot readability). The observed acceleration was close to $b$, but different values of $b$ result in different trajectories and thus different solution quality. Note that greater $b$ introduces deeper relaxation. Therefore, the value of $b$ should be set as a compromise between the quality and computational time of the LB. Nonetheless, the presented research confirms that the approach yields much faster convergence early on, and for $b = 5$ the inaccuracy caused by the relaxation is negligible.

**Aggregation.** Figure 7 shows trajectories for aggregated relaxation for $m = \{10, 50, 250\}$ (once again, the number of cases is limited for the readability of the figure). For $m = 10$ and $m = 50$, we observe that the aggregated relaxation results in much higher convergence speed. The trajectories have different shapes, but the acceleration can be approximated as being between 3 and 4 for $m = 50$ and between 6 and 10 for $m = 10$. It should also be noted that for $m = 10$ there is little loss in the LB and UB quality, and $m = 50$ provides even better LB than non-aggregated variant. On the other hand, for $m = 250$ the aggregated method converges much slower than the non-aggregated variant. This is because at $m = 250$, the $O(nm^2)$ complexity of the aggregated is larger than $O(nH)$ complexity of the non-aggregated method ($H = 2416$ for this instance). Thus, $m$ needs to be small enough. Interestingly, even seemingly large relaxation with $m = 10$ still allows us to obtain good quality results.

**8.3. Large-scale experiment.** Due to the number of methods, variables, stopping conditions and a limited length of this article, a comprehensive study of the

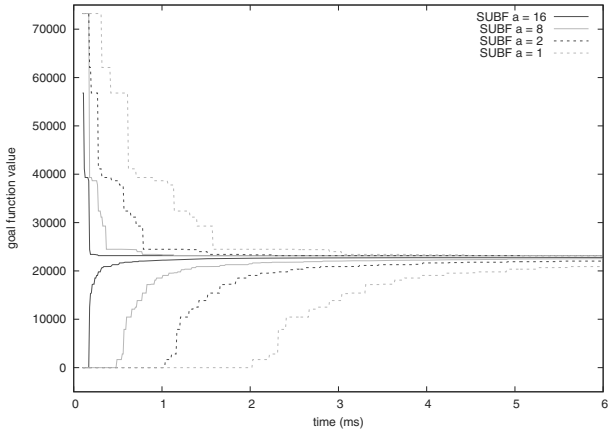


Fig. 5. Results of GCD scaling using SUBF and the illustrative instance for various values of $a$.

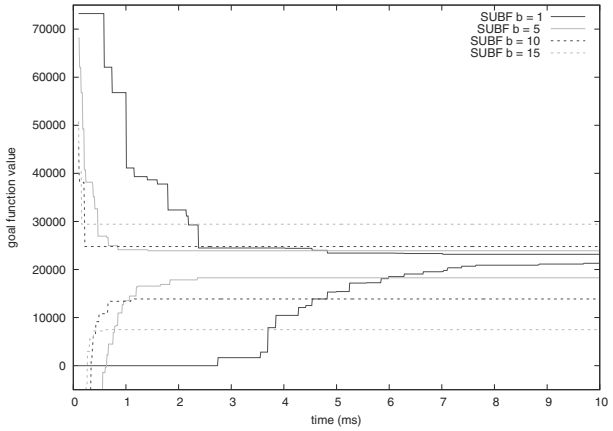


Fig. 6. Results of arbitrary scaling using SUBF and the illustrative instance for various values of $b$.

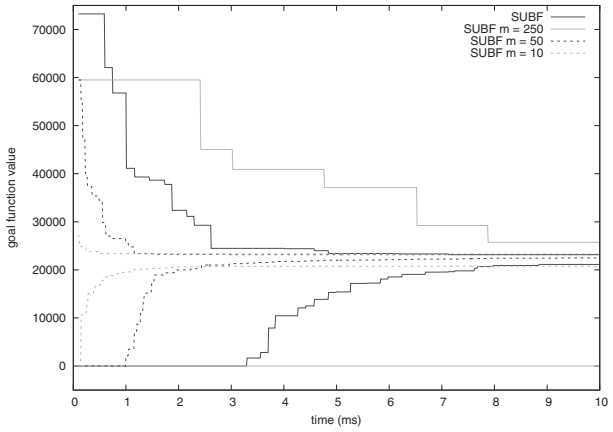


Fig. 7. Results of aggregated relaxation using SUBF and the illustrative instance for various values of $m$.

proposed methods is not possible here. Nonetheless, we carried out limited research on multiple problem instances. We considered three problem sizes $n \in \{30, 40, 50\}$ with 50 instances each (150 instances in total). To highlight the strengths and weaknesses of the methods, the instances were divided into two groups. For the first half instances, the $p_i$ values were multiplied by 16 as before, resulting in larger $H$ relative to the problem size. We term those instances "large $H$ instances." For the second half of the instances, the multiplication by 16 is skipped. Those are termed "small $H$ instances." The stopping condition was $0.1n$ milliseconds (except for the greedy heuristic).

Since multiple instances were used in this experiment, we cannot compare the obtained results directly. Instead we use percentage relative deviation (PRD). Let us start with upper bound. Given instance $I$ and algorithm $A$, we define $\mathrm{PRD}_{\mathrm{UB}}(A, I)$ as

$$\mathrm{PRD}_{\mathrm{UB}}(A, I) = 100\% \frac{A_{\mathrm{UB}}(I) - \mathrm{REF}_{\mathrm{UB}}(I)}{\mathrm{REF}_{\mathrm{UB}}(I)}, \qquad (68)$$

where $A_{\mathrm{UB}}(I)$ is the UB obtained by algorithm $A$ for instance $I$ and $\mathrm{REF}_{\mathrm{UB}}(I)$ is the UB obtained by the reference algorithm. In our case, we used the best of all tested algorithms as $\mathrm{REF}_{\mathrm{UB}}(I)$. In other words, $\mathrm{PRD}_{\mathrm{UB}}(A, I) = 20\%$ means that for instance $I$ algorithm $A$ obtained a 20% worse (larger) UB than the best tested algorithm. The PRD for the lower bound is defined similarly:

$$\mathrm{PRD}_{\mathrm{LB}}(A, I) = 100\% \frac{\mathrm{REF}_{\mathrm{LB}}(I) - A_{\mathrm{LB}}(I)}{\mathrm{REF}_{\mathrm{LB}}(I)}. \qquad (69)$$

As before, as $\mathrm{REF}_{\mathrm{LB}}(I)$ we chose the best LB obtained by all tested algorithms.

The summarized average $\mathrm{PRD}_{\mathrm{UB}}$ and $\mathrm{PRD}_{\mathrm{LB}}$ over all instances for each algorithm considered are shown in Tables 2 and 3, respectively. The complete raw result data is available on GitHub (Idzikowski, 2023). The most important results obtained by our newly proposed algorithms are shown in bold. The symbol "n/a" (not applicable) means that the algorithm does no provide this result.

Considering the large $H$ instances, we observe that arbitrary scaling managed to greatly improve the SUBF and SUBR methods. A similar situation, though to a lesser degree, can be observed for aggregated relaxation. The regular variants of SUBF and SUBR perform worse, but still generally outperform the remaining methods. It should also be noted that SUBR yields better results than SUBF. As for the m-SA method, it performs similarly to SUBF and SUBR as far as the LB is concerned, but is worse when it comes the UB. Finally, the SUBH method performs the worst and, interestingly, is not improved by any arbitrary scaling and aggregated

Table 2. Summary of large-scale experiments for large $H$ instances.

| Algorithm | $\mathrm{PRD}_{\mathrm{LB}}$ | $\mathrm{PRD}_{\mathrm{UB}}$ |
|---|---|---|
| SUBH | 80.1 | 127.9 |
| SUBH $m = H/50$ | 100.0 | 201.5 |
| SUBH $b = 3$ | 59.1 | 145.6 |
| SUBF | 84.9 | 35.8 |
| SUBF $b = 3$ | **2.6** | **0.4** |
| SUBF $m = H/50$ | **11.4** | **23.8** |
| SUBR | 60.7 | 42.2 |
| SUBR $b = 3$ | **1.1** | **2.2** |
| SUBR $m = H/50$ | **12.9** | **40.0** |
| m-SA | 74.4 | 96.6 |
| greedy | n/a | 216.3 |
| $\mathrm{DS}_{\mathrm{swap}}$ | n/a | 4.6 |
| $\mathrm{DS}_{\mathrm{adj}}$ | n/a | 69.3 |
| $\mathrm{DS}_{\mathrm{ins}}$ | n/a | 8.6 |

Table 3. Summary of large-scale experiments for small $H$ instances.

| Algorithm | $\mathrm{PRD}_{\mathrm{LB}}$ | $\mathrm{PRD}_{\mathrm{UB}}$ |
|---|---|---|
| SUBH | 38.6 | 67.0 |
| SUBH $m = H/50$ | 100.0 | 211.2 |
| SUBH $b = 3$ | 133.7 | 270.0 |
| SUBF | 0.0 | 0.1 |
| SUBF $b = 3$ | 30.9 | 12.1 |
| SUBF $m = H/50$ | 213.3 | 256.7 |
| SUBR | 2.4 | 1.0 |
| SUBR $b = 3$ | 37.0 | 11.7 |
| SUBR $m = H/50$ | 163.5 | 244.1 |
| m-SA | **8.7** | **3.0** |
| greedy | n/a | 221.9 |
| $\mathrm{DS}_{\mathrm{swap}}$ | n/a | 4.8 |
| $\mathrm{DS}_{\mathrm{adj}}$ | n/a | 71.0 |
| $\mathrm{DS}_{\mathrm{ins}}$ | n/a | 8.6 |

constraints techniques. We also note that the arbitrary scaling approach with SUBR and SUBF outperformed all variants of the DS algorithm.

In the case of small $H$ instances, the situation changes, as arbitrary scaling fails to improve the base variants of methods. This is due to the fact that for small values of $p_i$ the scaling introduces a considerable approximation error. Similarly, the aggregation relaxation also leads to considerably worse results, as for small $H$ the resulting number $m$ is too small to provide a good approximation. Of course, one could choose higher $m$ to reduce the error, but one has to be careful as this increases the computation time. Overall, the base variant of SUBF achieved the best performance for small $H$ instances, followed by SUBR. We also note that m-SA yielded very good results and outperformed the DS algorithm, proving that LR-based metaheuristics are a viable alternative to the subgradient method and classic approaches.

## 9. Conclusions

In this paper, we presented several ideas for using Lagrangian relaxation to solve the total weighted tardiness scheduling problem. The general approach can be easily extended to cover single-machine scheduling with ready times, job precedence constraints, any non-decreasing cost function. Since more complex scheduling problems, for example, flow-shop, job-shop, open-shop, can be introduced by the precedence among operations, the proposed ideas can be extended to a wide class of scheduling problems.

Regarding the basic relaxation approach, we presented an arbitrary scaling technology together with a discussion of its properties and correctness. With computer experiments we illustrated that this approach can greatly improve the convergence rate of the subgradient method. The results also showed how scaling affects their quality and that it is possible to reduce the computation time with little loss in quality.

We also proposed a novel Lagrangian relaxation approach to the problem considered based on constraint aggregation. This approach is adjustable, allowing one to find a compromise between the relaxation error and the algorithm running time. The experiments indicated that, compared with the original relaxation method, this approach allows obtaining similar quality results in much shorter time. Regarding the basic relaxation approach, we proposed a solution method based on a local-search metaheuristic. The results showed that it is a viable alternative to the subgradient method and exhibits faster convergence rate early on, compared with the subgradient method. We believe that further research in applying metaheuristics to this problem would result in an algorithm capable of outperforming the subgradient method.

Finally, in our research we focused on reducing the computation time of the solution methods. The computer experiments indicated that the proposed method is capable of obtaining good quality results in a few milliseconds for problem instances of 30 to 50 jobs. As a result, the proposed approach could be employed as a subroutine in other solving methods.

## References

Ali, S.M., Fathollahi-Fard, A.M., Ahnaf, R. and Wong, K.Y. (2023). A multi-objective closed-loop supply chain under uncertainty: An efficient Lagrangian relaxation reformulation using a neighborhood-based algorithm, *Journal of Cleaner Production* **423**: 138702, DOI: 10.1016/j.jclepro.2023.138702.

Araújo, C.V.D., de Souza, C.C. and Usberti, F.L. (2024). Lagrangian relaxation for maximum service in multicast routing with QoS constraints, *International Transactions in Operational Research* **31**(1): 140–166, DOI: 10.1111/itor.13200.

Beasley, J.E. (1990). *OR-Library*, `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

Bilge, Ü., Kurtulan, M. and Kıraç, F. (2007). A tabu search algorithm for the single machine total weighted tardiness problem, *European Journal of Operational Research* **176**(3): 1423–1435, DOI: 10.1016/j.ejor.2005.10.030.

Bragin, M.A., Luh, P.B., Yan, J.H., Yu, N. and Stern, G.A. (2015). Convergence of the surrogate Lagrangian relaxation method, *Journal of Optimization Theory and Applications* **164**(1): 173–201, DOI: 10.1007/s10957-014-0561-3.

Butt, A.A. and Collins, R.T. (2013). Multi-target tracking by Lagrangian relaxation to min-cost network flow, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, USA*, pp. 1846–1853, DOI: 10.1109/CVPR.2013.241.

Chen, H. and Luh, P.B. (2003). An alternative framework to Lagrangian relaxation approach for job shop scheduling, *European Journal of Operational Research* **149**(3): 499–512, DOI: 10.1109/CDC.1999.832909.

Congram, R.K., Potts, C.N. and van de Velde, S.L. (2002). An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem, *INFORMS Journal on Computing* **14**(1): 52–67, DOI: 10.1287/ijoc.14.1.52.7712.

Crauwels, H., Potts, C.N. and Van Wassenhove, L.N. (1998). Local search heuristics for the single machine total weighted tardiness scheduling problem, *INFORMS Journal on Computing* **10**(3): 341–350, DOI: 10.1287/ijoc.10.3.341.

Cui, H. and Luo, X. (2017). An improved Lagrangian relaxation approach to scheduling steelmaking-continuous casting process, *Computers & Chemical Engineering* **106**: 133–146, DOI: 10.1016/j.compchemeng.2017.05.026.

Everett, H. (1963). Generalized Lagrange multiplier method for solving problems of optimum allocation of resources, *Operations Research* **11**(3): 399–417, DOI: 10.1287/opre.11.3.399.

Fisher, M.L. (1976). A dual algorithm for the one-machine scheduling problem, *Mathematical Programming* **11**(1): 229–251, DOI: 10.1007/BF01580393.

Fisher, M.L. (2004). The Lagrangian relaxation method for solving integer programming problems, *Management Science* **50**(12_supp): 1861–1871, DOI: 10.1287/mnsc.1040.0263.

Fu, Y.-M. and Diabat, A. (2015). A Lagrangian relaxation approach for solving the integrated quay crane assignment and scheduling problem, *Applied Mathematical Modelling* **39**(3–4): 1194–1201, DOI: 10.1016/j.apm.2014.07.006.

Gaudioso, M., Gorgone, E., Labbé, M. and Rodríguez-Chía, A.M. (2017). Lagrangian relaxation for SVM feature selection, *Computers & Operations Research* **87**: 137–145, DOI: 10.1016/j.cor.2017.06.001.

Gocgun, Y. and Ghate, A. (2012). Lagrangian relaxation and constraint generation for allocation and advanced scheduling, *Computers & Operations Research* **39**(10): 2323–2336, DOI: 10.1016/j.cor.2011.11.017.

Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnoy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics* **5**: 287–326, DOI: 10.1016/S0167-5060(08)70356-X.

Hajibabaei, M. and Behnamian, J. (2023). Fuzzy cleaner production in assembly flexible job-shop scheduling with machine breakdown and batch transportation: Lagrangian relaxation, *Journal of Combinatorial Optimization* **45**(5): 112, DOI: 10.1007/s10878-023-01046-1.

Held, M., Wolfe, P. and Crowder, H.P. (1974). Validation of subgradient optimization, *Mathematical Programming* **6**(1): 62–88, DOI: 10.1007/BF01580223.

Huang, D., Wang, Y., Jia, S., Liu, Z. and Wang, S. (2023). A lagrangian relaxation approach for the electric bus charging scheduling optimisation problem, *Transportmetrica A: Transport Science* **19**(2): 2023690, DOI: 10.1080/23249935.2021.2023690.

Idzikowski, R. (2023). *LagrangianRelaxation*, Code and experimental results, `https://github.com/ridzikowski/LagrangianRelaxation`.

Koulamas, C. (2010). The single-machine total tardiness scheduling problem: Review and extensions, *European Journal of Operational Research* **202**(1): 1–7, DOI: 10.1016/j.ejor.2009.04.007.

Lawler, E.L. (1977). A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness, *Annals of Discrete Mathematics* **1**: 331–342, DOI: 10.1016/S0167-5060(08)70742-8.

Lenstra, J.K., Kan, A.R. and Brucker, P. (1977). Complexity of machine scheduling problems, *Annals of Discrete Mathematics* **1**: 343–362, DOI: 10.1016/S0167-5060(08)70743-X.

Liu, X., Wang, W., Chen, X., Sterna, M. and Blazewicz, J. (2023). Exact approaches to late work scheduling on unrelated machines, *International Journal of Applied Mathematics and Computer Science* **33**(2): 285–295, DOI: 10.34768/amcs-2023-0021.

Matsuo, H., Juck Suh, C. and Sullivan, R.S. (1989). A controlled search simulated annealing method for the single machine weighted tardiness problem, *Annals of Operations Research* **21**(1): 85–108, DOI: 10.1007/BF02022094.

Nishi, T., Hiranaka, Y. and Inuiguchi, M. (2010). Lagrangian relaxation with cut generation for hybrid flowshop scheduling problems to minimize the total weighted tardiness, *Computers & Operations Research* **37**(1): 189–198, DOI: 10.1016/j.cor.2009.04.008.

Nowak, M.P. and Römisch, W. (2000). Stochastic Lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty, *Annals of Operations Research* **100**(1–4): 251–272, DOI: 10.1023/A:1019248506301.

Potts, C.N. and Van Wassenhove, L.N. (1985). A branch and bound algorithm for the total weighted tardiness problem, *Operations Research* **33**(2): 363–377, DOI: 10.1287/opre.33.2.363.

Potts, C. and Van Wassenhove, L.N. (1991). Single machine tardiness sequencing heuristics, *IIE Transactions* **23**(4): 346–354, DOI: 10.1080/07408179108963868.

Simanchev, R.Y. and Urazova, I. (2023). Integer models for the total weighted tardiness problem on a single machine, *International Conference on Mathematical Optimization Theory and Operations Research, Ekaterinburg, Russia*, pp. 176–187, DOI: 10.1007/978-3-031-43257-6_14.

Song, M., Cheng, L. and Lu, B. (2024). Solving the multi-compartment vehicle routing problem by an augmented Lagrangian relaxation method, *Expert Systems with Applications A* **237**: 121511, DOI: 10.1016/j.eswa.2023.121511.

Song, M., Lu, B., Cheng, L. and Sun, C. (2023). Lagrangian relaxation-based decomposition approaches for the capacitated arc routing problem in the state-space-time network, *Transportation Letters* **15**(10): 1317–1336, DOI: 10.1080/19427867.2022.2148368.

Speckenmeyer, P., Hilmer, C., Rauchecker, G. and Schryen, G. (2023). Parallel branch-and-price algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times, *SSRN:* 4537436, (preprint).

Zakharova, Y. (2023). Hybrid evolutionary algorithm with optimized operators for total weighted tardiness problem, *International Conference on Mathematical Optimization Theory and Operations Research, Ekaterinburg, Russia*, pp. 224–238, DOI: 10.1007/978-3-031-35305-5_15.

Zhang, C., Gao, Y., Yang, L., Gao, Z. and Qi, J. (2020). Joint optimization of train scheduling and maintenance planning in a railway network: A heuristic algorithm using Lagrangian relaxation, *Transportation Research B: Methodological* **134**: 64–92, DOI: 10.1016/j.trb.2020.02.008.

Zhao, Q. and Yuan, J. (2023). Single-machine primary-secondary scheduling with total tardiness being the primary criterion, *Journal of Scheduling* **27**(3): 1–10, DOI: 10.1007/s10951-023-00793-7.

Zhou, Y. and Lee, G.M. (2017). A Lagrangian relaxation-based solution method for a green vehicle routing problem to minimize greenhouse gas emissions, *Sustainability* **9**(5): 776, DOI: 10.3390/su9050776.

**Jarosław Rudy** received his PhD degree in computer science from the Faculty of Electronics, Wrocław University of Science and Technology, Poland, in 2016. He works as an assistant professor at the Department of Control Systems and Mechatronics of the Wrocław University of Science and Technology. He is the author or a co-author of over 40 publications, mainly in the fields of discrete optimization, operations research and job scheduling. His other research interests include parallel computing, computational complexity, metaheuristics and vehicle routing. ORCID: 0000-0003-1095-6041.

**Radosław Idzikowski** is employed as an assistant professor in the Department of Control Systems and Mechatronics at the Faculty of Information and Communication Technology of the Wrocław University of Science and Technology. He is a member of the international MOCOS (Modelling Coronavirus Spread) team specializing in modeling the COVID-19 epidemic. In his research, he focuses on optimization algorithms and artificial intelligence methods for solving task scheduling and transport problems. His doctoral dissertation was dedicated to various variants of flow shop problem with time couplings. ORCID: 0000-0001-5232-1998.

**Czesław Smutnicki** (PhD: 1981, DSc: 1998, full prof.: 2005) is a full professor at the Wrocław University of Science and Technology. He has supervised six completed PhD thesis and has published 200+ manuscripts, including five books and textbooks. His number of citations in SCI is 1800+. He has cooperated in and leads a few research grants. His research interests include optimization, scheduling, discrete systems and transport. ORCID: 0000-0003-4640-5364

**Zbigniew A. Banaszak** holds a full professorial position at the Koszalin University of Technology, Poland. Since 1979 he has been the principal investigator or a co-investigator for 40 research projects at the Wrocław University of Technology, Kuwait University, Hull University and the Warsaw University of Technology, as well as the Systems Research Institute of the Polish Academy of Sciences. He has supervised 17 completed PhD theses and has published over 500 manuscripts, including 19 books and text books. His research interests are in the areas of the discrete dynamic systems theory, decision support systems, constraints programming driven planning and scheduling with application to multimodal supply chine networks. ORCID: 0000-0001-7219-3903.

**Grzegorz Bocewicz** is an associate professor at the Faculty of Electronics and Computer Science of the Koszalin University of Technology, Poland. He obtained his PhD and DSc degrees in computer sciences from the Wrocław University of Technology, Poland, in 2007 and 2014, respectively. His research interests are in the areas of the modeling and design of decision support systems, methods of advanced planning and scheduling, modeling and analyzing of systems of concurrent cyclic processes, and projects on portfolio prototyping under uncertain constraints. He is a co-author of over 70 journal articles, three books, 50 books chapters, and 60 peer reviewed conference papers. ORCID: 0000-0002-5181-2872.

## Appendix

The correctness of (35) can be shown as follows. First, let us notice that $L'(S', u')$ is a scaled version of the same problem, so (17) still applies. Therefore,

$$L'(S',u') = \sum_{i=1}^{n}\Big(w_i T_i' + \sum_{r=S_i'+1}^{S_i'+p_i'} u_r'\Big) - \sum_{r=1}^{H'} u_r'. \quad \text{(A1)}$$

We shall now transform this formula to use symbols from the nonscaled problem (e.g., $p_i$ instead of $p_i'$, and so on) and extension $\hat{u}_t$ instead of $u_r'$. Let us start by proving that

$$\sum_{r=1}^{H'} u_r' = \frac{1}{a}\sum_{t=1}^{H}\hat{u}_t. \quad \text{(A2)}$$

Summing (36) for fixed $r$ and the whole given range of $t$, we have

$$\sum_{t=(r-1)a+1}^{ra}\hat{u}_t = a u_r', \quad r = 1,2,\dots H'. \quad \text{(A3)}$$

Then, applying (A3) and the transformations (32), we get

$$\begin{aligned}\sum_{r=1}^{H'} u_r' &= \frac{1}{a}\sum_{r=1}^{H'} a u_r' \\ &= \frac{1}{a}\sum_{r=1}^{\frac{H}{a}}\sum_{t=(r-1)a+1}^{ra}\hat{u}_t \\ &= \frac{1}{a}\Big(\sum_{t=1}^{a}\hat{u}_t + \sum_{t=a+1}^{2a}\hat{u}_t + \sum_{t=2a+1}^{3a}\hat{u}_t + \dots \\ &\quad + \sum_{t=H-a+1}^{H}\hat{u}_t\Big) = \frac{1}{a}\sum_{t=1}^{H}\hat{u}_t. \end{aligned} \quad \text{(A4)}$$

Next, we shall prove that

$$\sum_{i=1}^{n}\Big(w_i T_i' + \sum_{r=S_i'+1}^{S_i'+p_i'} u_r'\Big) = \sum_{i=1}^{n}\Big(w_i\frac{T_i}{a} + \frac{1}{a}\sum_{t=S_i+1}^{S_i+p_i}\hat{u}_t\Big). \quad \text{(A5)}$$

We start with the decomposition,

$$\begin{aligned}&\frac{1}{a}\sum_{r=S_i'+1}^{S_i'+p_i'} a u_r' \\ &= \frac{1}{a}\Big(a u_r'|_{r=S_i'+1} + a u_r'|_{r=S_i'+2} \\ &\quad + \dots + a u_r'|_{r=S_i'+p_i'}\Big). \end{aligned} \quad \text{(A6)}$$

Similarly as before, adding (36) for a given range of $t$ and for each $r$ specified in (A6) with the use of the transformations (32), we obtain the sequence of equations

$$\sum_{t=S_i+1}^{S_i+a}\hat{u}_t = a u_r'|_{r=S_i'+1}, \quad \text{(A7)}$$

$$\sum_{t=S_i+a+1}^{S_i+2a}\hat{u}_t = a u_r'|_{r=S_i'+2}, \quad \text{(A8)}$$

$$\vdots \quad \text{(A9)}$$

$$\sum_{t=S_i+p_i-a+1}^{S_i+p_i}\hat{u}_t = a u_r'|_{r=S_i'+p_i'}. \quad \text{(A10)}$$

Substituting (A7) ... (A10) into (A6), we obtain

$$\begin{aligned}\frac{1}{a}\sum_{r=S_i'+1}^{S_i'+p_i'} a u_r' &= \frac{1}{a}\Big(\sum_{t=S_i+1}^{S_i+a}\hat{u}_t + \sum_{t=S_i+a+1}^{S_i+2a}\hat{u}_t \\ &\quad + \dots \sum_{t=S_i+p_i-a+1}^{S_i+p_i}\hat{u}_t\Big) \\ &= \frac{1}{a}\sum_{t=S_i+1}^{S_i+p_i}\hat{u}_t. \end{aligned} \quad \text{(A11)}$$

With this and $T_i' = T_i/a$, we transformed (A1) into

$$\sum_{i=1}^{n}\Big(w_i\frac{T_i}{a} + \frac{1}{a}\sum_{t=S_i+1}^{S_i+p_i}\hat{u}_t\Big) - \frac{1}{a}\sum_{t=1}^{H}\hat{u}_t, \quad \text{(A12)}$$

which, due to (17), is equal to $L(S,\hat{u})/a$.

Received: 17 October 2023
Revised: 18 February 2024
Re-revised: 15 March 2024
Accepted: 18 March 2024