Metaheuristic-Based Lagrangian Relaxation for Total Weighted Tardiness Minimization

Czesław Smutnicki[1], Jarosław Rudy[2], Radosław Idzikowski[2(✉)], and Wojciech Bożejko[2]

[1] Department of Computing Engineering, Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
czeslaw.smutnicki@pwr.edu.pl

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

Abstract. In this paper, a Lagrangian relaxation approach to the problem of minimization of the total weighted tardiness is presented. The approach can be used both to solve the original problem as well as to obtain a high-quality lower bound for the branch-and-bound method. The problem was further decomposed into upper and lower problems. For the upper problem, an approach using a local-search metaheuristic is proposed instead of a subgradient method from the literature. Several variants of both methods are then implemented and compared through a numerical example using OR library problem instances. The results prove the viability of the metaheuristic approach as with little calibration it is able to match or outperform a refined subgradient method.

Keywords: total weighted tardiness · lagrangian relaxation · metaheuristics · lower bound

1 Introduction

The problem of minimizing the weighted tardiness of jobs denoted $1||\sum w_i T_i$ in Graham notation, is one of the most commonly considered single-machine scheduling problems in both theory and practice. [1,16]. Due to its NP-hardness, Branch-and-Bound (B&B) method remains one of the most popular approaches to this problem [17]. B&B is based on computing Upper (UB) and Lower Bounds (LB). Computing UB is relatively easy since any feasible solution to $1||\sum w_i T_i$ can serve as such. Computing the value of LB is harder in practice, especially for problems such as $1||\sum w_i T_i$. In this paper, we propose an approach based on Lagrangian relaxation with the use of a hybrid algorithm and metaheuristics.

M. Pawelczyk et al. (Eds.): PCC 2023, LNNS 709, pp. 98–107, 2023.
https://doi.org/10.1007/978-3-031-35173-0_10

2 Primal Problem

The primal problem $1||\sum w_i T_i$ is as follows. We have a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of jobs, with processing time $p_i > 0$, deadline $d_i > 0$ and weight $w_i > 0$ for each job i, to process on the single machine. The aim is to determine job starting times $S = (S_1, S_2, \ldots, S_n)$, as to minimize the total weighted tardiness:

$$\min_S \sum_{i=1}^{n} w_i [S_i + p_i - d_i]^+ = \min_S \sum_{i=1}^{n} w_i T_i, \tag{1}$$

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:

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

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 processing order (permutation) of jobs $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, where $\pi(i)$ is the job that will be processed as i-th. As such, for a given processing order π the schedule S is computed recursively in time $O(n)$ as follows:

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

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

3 Lagrangian Relaxation and Upper Bound

Lagrangian relaxation is based on relaxing some of the constraints of the primal problem and including them into objective function as a penalty [7]. Due to its generality, this approach had been applied to a wide range of optimization problems. In job scheduling examples include flowshop [6,13], jobshop [4], stochastic scheduling [14], crane assignment [8] and industrial process such as casting [5]. Aside from scheduling, Lagrangian relaxation had been used in vehicle routing [12], resource allocation [10], min-flow problems [3] and classification [9].

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 processing times p_i are integers. 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, $H = \sum_{i=1}^{n} p_i$. With this, for a given schedule S, we can define the set of jobs processed in time interval $[t-1, t]$:

$$\mathcal{I}_t(S) \stackrel{\text{def}}{=} \{i \in \mathcal{N} : t > S_i \geq t - p_i\}, \tag{5}$$

and the number of jobs processed in that interval:

$$g_t(S) \stackrel{\text{def}}{=} |\mathcal{I}_t(S)|. \tag{6}$$

The problem from Sect. 2 can be stated as a non-linear optimization case:

$$\min_{S} \sum_{i=1}^{n} w_i T_i, \tag{7}$$

$$g_t(S) = 1, \qquad t = 1, 2, \ldots, H, \tag{8}$$

$$0 \le S_i \le H - p_i, \qquad i = 1, 2, \ldots, n. \tag{9}$$

Constraint (8) ensures that exactly one job is processed in time interval $[t-1, t]$. Constraint (9) is stated directly by introducing the following set:

$$\mathcal{S} \stackrel{\text{def}}{=} \{S = (S_1, S_2, \ldots, S_n) : 0 \le S_i \le H - p_i, \;\; i = 1, 2, \ldots, n\}. \tag{10}$$

An example of how function $g_t(S)$ works is shown in Fig. 1. In the upper part, there is an exemplary schedule S for 5 jobs with $\sum_{p_i} = H = 21$. Since we relaxed the problem, the jobs can overlap. The lower part shows the plot of $g_t(S)$ with its value for a given t corresponding to the number of jobs processed at t in the upper part. In the original problem, only one job can be processed at the same time, so $g_t(S)$ would be equal to 1 for $t = [0, H]$ (dashed line). In the relaxed problem jobs can overlap, giving us values of $g_t(S)$ different than 1.

To take (8) into account, we introduce dual variables $u \stackrel{\text{def}}{=} (u_1, u_2, \ldots, u_H)$, where u_t is assigned for a fixed t. We can perceive u_t as the cost of using the machine in the interval $[t-1, t]$. We obtain the following Lagrange function:

$$L(S, u) \stackrel{\text{def}}{=} \sum_{i=1}^{n} w_i T_i + \sum_{t=1}^{H} u_t (g_t(S) - 1). \tag{11}$$

Let us notice that in primal problem $1||\sum w_i T_i$, for each left-shifted *feasible* schedule S and all $t = 1, 2, \ldots, H$ it must hold that $g_t(S) = 1$. In such a case Eq. (11) is equal to the objective function $\sum_{i=1}^{n} w_i T_i$ for schedule S.

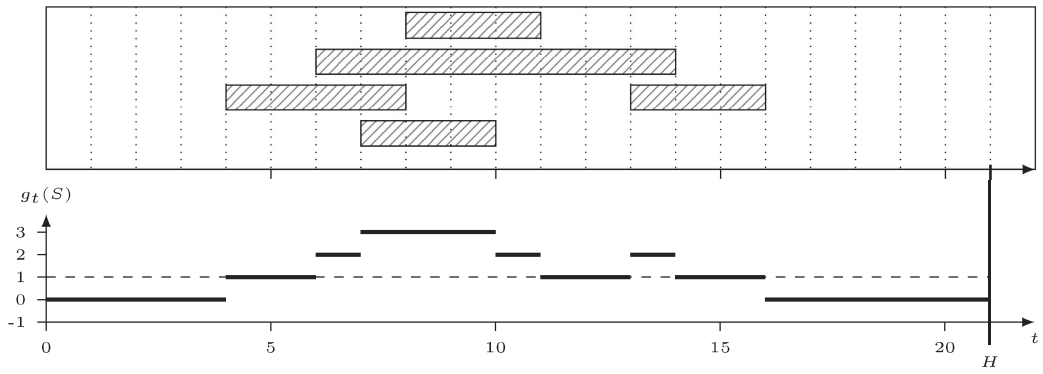


Fig. 1. Exemplary relaxed schedule and $g_t(S)$ function corresponding to it.

We now transform function (11) into a more convenient form:

$$L(S,u) = \sum_{i=1}^{n}(w_i T_i + \sum_{t=S_i+1}^{S_i+p_i} u_t) - \sum_{t=1}^{H} u_t = \sum_{i=1}^{n} L_i(S_i,u) - U_H, \tag{12}$$

$$L_i(S_i,u) \stackrel{\text{def}}{=} w_i T_i + \sum_{t=S_i+1}^{S_i+p_i} u_t = w_i T_i + U_{S_i+p_i} - U_{S_i} \tag{13}$$

$$U_t = \sum_{s=1}^{t} u_s, \qquad t = 1,2,\ldots,H. \tag{14}$$

Let S^* be an optimal solution to the primal problem (7)–(9) and $W(u)$ be an optimal solution to $\min_S L(S,u)$ for a fixed u. Then for all u it holds that:

$$\sum_{i=1}^{n} w_i[S_i^* + p_i - d_i]^+ = \sum_{i=1}^{n} w_i[S_i^* + p_i - d_i]^+ + \sum_{t=1}^{H} u_t(g_t(S^*) - 1) \geq \tag{15}$$

$$\geq \min_{S\in\mathcal{S}} L(S,u) \stackrel{\text{def}}{=} W(u). \tag{16}$$

Thus, $W(u)$ is a lower bound (LB) for the objective function (7) due to the duality principle. We are to maximize this LB, thus:

$$\max_u W(u) = \max_u \min_S L(S,u). \tag{17}$$

The above model can be used to solve the primal problem (7)–(9): if for some u we find S^* from the minimization of $\min_S L(S,u) = L(S^*,u)$ such that $g_t(S^*) = 1$, $t = 1,2,\ldots,H$, then S^* is an optimal solution to problem (7)–(9).

Problem (17) can be decomposed into two problems: $\max_u W(u)$, henceforth called the upper problem, and $\min_S L(S,u)$, henceforth called the lower problem. The lower problem is usually solved multiple times for different values of u. To solve the upper problem we can use an inexact method (e.g. a heuristic), but the lower problem has to be solved to optimality to ensure the resulting values is LB as per (16). We will discuss the approaches to both problems.

4 The Lower Problem

By applying (10)–(13) for a fixed u we obtain:

$$W(u) = \min_{S\in\mathcal{S}} L(S,u) = \sum_{i=1}^{n} \min_{0\leq S_i\leq H-p_i} L_i(S_i,u) - U_H = \sum_{i=1}^{n} V_i(u) - U_H \tag{18}$$

$$V_i(u) \stackrel{\text{def}}{=} \min_{0\leq S_i\leq H-p_i} L_i(S_i,u) = \min_{S_i=0,1,\ldots,H-p_i} L_i(S_i,u). \tag{19}$$

For a fixed u each problem (19) can be solved by directly evaluating all possible starting times S_i. Since formula (14) can be stated in a recursive form $U_t = U_{t-1} + u_t$, $t = 1,2,\ldots,H$, $U_0 = 0$, computing value U_t can be done in time $O(H)$. Thus, determining (13) with the use of (19) requires time $O(H)$, while determining $W(u)$ through formula (18) for a fixed u requires time $O(nH)$.

5 The Upper Problem

Problem (17) is a continuous optimization problem with a non-linear (piecewise linear in fact) weakly convex function that is not differentiable at "interval junction" points and with a high size of vector u. Formally, it can be reformulated as linear programming, but the resulting size makes such reformulation infeasible. For example, for $H = 1000$ and $n = 100$ the expected formulation has 10^3 variables and 10^{300} constraints. Due to non-differentiability several alternative approaches are proposed: (1) subgradient method, (2) local-search metaheuristic, (3) a hybrid method. We will discuss those approaches below.

5.1 Subgradient Method

Subgradient (SUB) method is often applied to Lagrangian relaxation problems [11]. Briefly speaking, SUB can be perceived as an iterative method that produces a sequence of vectors $u^0, u^1, u^2, \ldots$ as per the following formula:

$$u_t^{k+1} = u_t^k + \alpha^k (g_t(S^k) - 1), \quad t = 1, 2, \ldots, H, \tag{20}$$

where α^k is the length of step in k-th iteration while S^k is an optimal solution to the lower problem in k-th iteration, i.e. $\min_S L(S, u^k) = L(S^k, u^k)$. The choice of sequence α^k heavily affects the stability, convergence, and convergence speed of the method. In theory, the convergence of LB to the value of 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 a 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. Lack of growth of $W(u^k)$ for $k = 1, 2, \ldots$, might be caused by "zig-zagging" phenomenon accompanied by α^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:

$$\alpha^k = \gamma^k \frac{W(u^k) - W^*}{||g(S^k) - 1||^2}, \tag{21}$$

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

$$||g(S^k) - 1||^2 = \sum_{t=1}^{H} (g_t(S^k) - 1)^2. \tag{22}$$

For α^k given by formula (21) convergence was proven. In practice, due to W^* being unknown, provisory value UB^k is used, even though it does not guarantee convergence. It is advised to update UB^k in each iteration k by using an *auxiliary* heuristic. One of the simplest approaches is computing the value of the objective function based on (3)–(4) for permutation π obtained by ordering all jobs according to the non-decreasing value of S_i^k. It is possible to use more complex methods (i.e. metaheuristics) instead, resulting in a better approximation of 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 [2].

5.2 Local-Search Metaheuristic

Our aim is to solve problem $\max_u W(u)$ using metaheuristics and to assess the feasibility of this approach. To our knowledge such approach does not exist in the literature. We will propose a variant of SA called next as *modified Simulated Annealing* (m-SA) method, though many other metaheuristics can be used.

The method m-SA produces a vector sequence $u^0, u^1, \ldots$ similar to a regular 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 as:

$$N(u^k) = \{u = (u_1, \ldots u_H) : \ u_t \in [u_t^k - A, u_t^k + (n-1)A], \ t = 1, \ldots, H\} \quad (23)$$

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

$$\tilde{u}_t = u_t^k + Z_t \cdot (g_t(S^k) - 1), \ t = 1, 2, \ldots, H, \quad (24)$$

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

Temperature is reduced systematically according to the so-called cooling scheme, usually in every iteration. Several such schemes have been defined in the literature. In our case we began from the geometric scheme $T^{k+1} = \lambda T^k$, $k = 0, 1, \ldots$. The resulting algorithm has several tuning parameters, such as u^0, T^0, λ or the stop condition A similar drawback "necessity of tuning" has also been found in Boltzman scheme $T^{k+1} = T^k/(1 + \lambda T^k)$. A self-calibrating variant of the cooling scheme also exists and can be used to alleviate this issue.

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

5.3 Hybrid Approaches

Here we will discuss a few modifications to SUB and m-SA in order to improve their computation complexity and convergence. We will start with modification to model (21) which can include: (1) initializing u^0 with random values, (2) guided control of sequence γ^k, (3) random control of sequence γ^k. Option (1) 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 (2) and (3) to increase the speed of convergence of $W(u)$ to UB.

Guided control aims to automatically adjust γ^k to shorten the initial phase with unfavorable values $W(u)$. One such approach is to set $\gamma^0 \approx 2.0$ and then reduce γ^k by half if there was no improvement of $W(u^k)$ in the last 5 iterations. Another method uses the 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 σ and τ should be close to 1.

Random changes of γ^k can be seen as another variant of SA The approach introduces a small random perturbation to γ^k in (21). The resulting random variable γ^k has an average value change according to (21). Practice indicates that such randomness prevents the "zig-zag" pattern specific to SUB from occurring.

6 Numerical Experiment

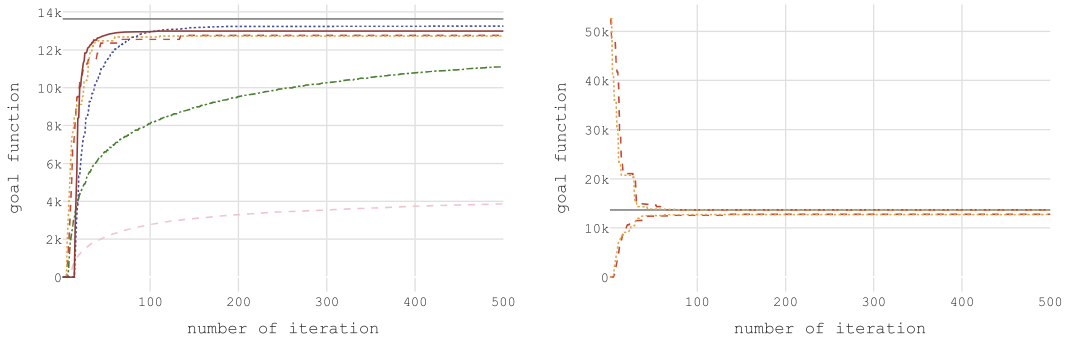
To illustrate the qualities and applicability of the above methods for computing LB and UB values, we have implemented them in a few variants as follows:

1. SUB_A – SUB with sequence $\alpha^k = \frac{c}{k}$, $c = 1$; pink dashed line in figures.
2. SUB_B – SUB with $\alpha^k = \frac{c}{\sqrt{k}}$, $c = 1$; green dash-dotted line in figures.
3. SUB_C – SUB with formulas (21)–(22) applied, where γ starts at 2 and is halved if $W(u)$ has not improved in 5 iterations; dotted blue line in figures.
4. SUB_D – SUB with formulas (21)–(22) applied, where γ^k is a random variable from uniform $[0.95; 1.05]$ distribution; brown solid line in figures.
5. SA_A – m-SA with $u^0 = \mathbf{u}$, $\lambda = 0.99$, $T^0 = 1000$, dashed red line in figures.
6. SA_B – m-SA with $u^0 = \mathbf{u}$, $\lambda = 0.9$, $T^0 = 100$; dotted orange line in figures.

Initially elements of $\mathbf{u}$ are set to random values from interval $u_t \in [-1.5...3.5]$, $t = 1, \ldots, H$. All methods were coded in `Julia` 1.7. The experiments were run on a machine with Intel Xeon CPU E5-2680 v3 12-core processor with 2.5 GHz clock and 64 GB of RAM under Windows 10. To allow comparison, all plots were generated from the same problem instance with $n = 30$, generated using the method from OR-library [15]. Values p_i, w_i and d_i were random integers from intervals $[1; 100]$, $[1; 10]$ and $[0.2; 0.6]$ respectively. The horizontal line in all figures represents the optimal solution value for the instance.

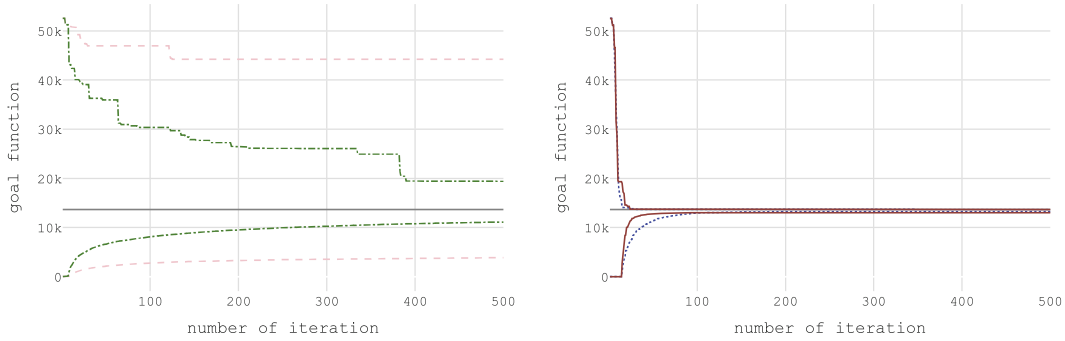
Use of the harmonic sequence (Fig. 5a) for α in the SUB method has guaranteed convergence, but in practice, this approach is very slow, ultimately leaving a large gap between LB and UB (also see Fig. 2a and 3a). The choice of the constant c is also troublesome. Decreasing the rate of α convergence to zero (Fig. 5b) usually improves the results (again see Fig. 2a and 3a).

Despite no theoretical convergence proof, the more complex model using γ, $W(u)$ and UB (see Fig. 5c and 5d) is behaving much more stable, providing better values LB and UB (also see Fig. 2a and 3b). Curiously, the randomly oscillating γ approach appears significantly better than its deterministic counterpart (Fig. 3b).



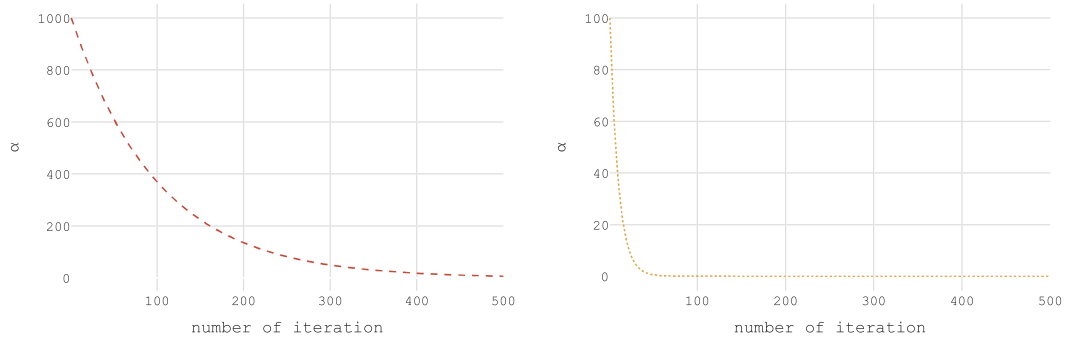
(a) LB for all methods (b) LB and UB for SA_A and SA_B

Fig. 2. Convergence of described methods



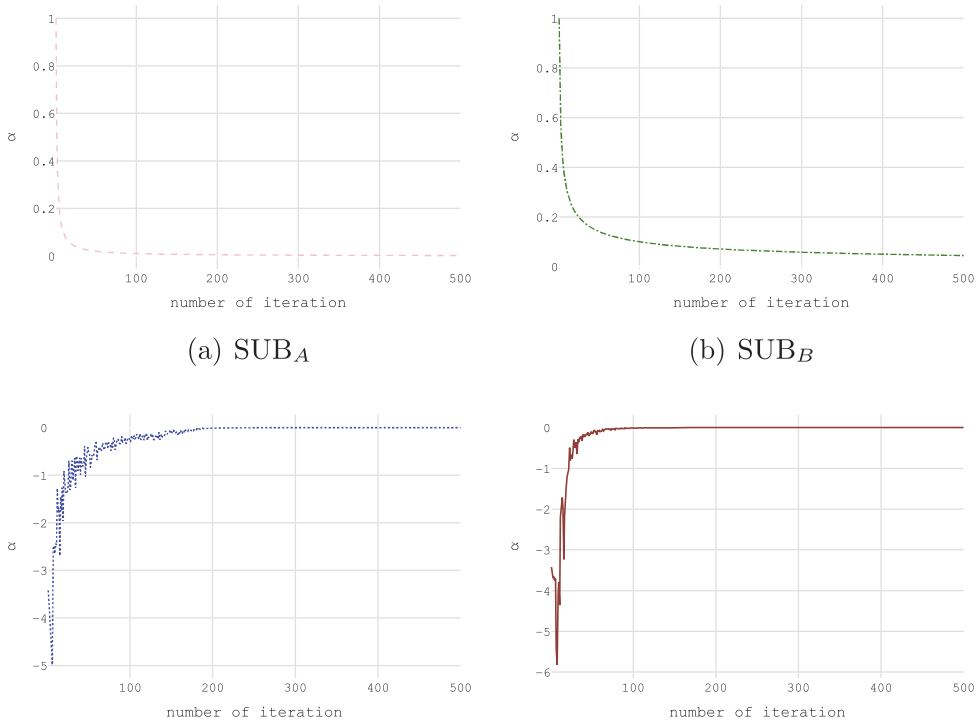
(a) Comparison of SUB_A and SUB_B (b) Comparison SUB_C and SUB_D

Fig. 3. Convergence of UB i LB for SUB variants.



(a) SA_A (b) SA_B

Fig. 4. Cooling schemes in SA method.



(a) SUB_A (b) SUB_B

(c) SUB_C (d) SUB_D

Fig. 5. Changes of α value in SUB variants.

The behavior varied from instance to instance, but the observations shown above appear typical. The presented example of using SA confirms that the idea of using locals-search metaheuristics for this task instead of the SUB method is promising. The tested example shows a good convergence speed compared to the SUB method (see Fig. 2a and 2b), despite the fact that next to no tuning for SA was performed. Thus, with proper tuning, it should be possible to improve SA further. Moreover, self-tuning can be applied to SA, removing the tedious process that remains so troublesome and time-consuming for the SUB method.

7 Conclusions and Future Work

An approach to total weighted tardiness minimization using Lagrange relaxation was shown to (1) obtain a lower bound for branch-and-bound and (2) solve the original problem. A few algorithm variants based on subgradient and simulated annealing methods were proposed. The examples shown indicated: (1) the effectiveness of the considered approach, (2) the competitiveness of the simulated annealing, (3) difficulties in tuning for the subgradient method.

Due to promising results, future work on this topic encompasses three main directions. The first direction seeks to confirm the quality of the results on a larger set of literature benchmark instances with a larger number of jobs. The second direction aims to improve the tuning of the proposed methods,especially

the hybrid ones. The third direction aims to decrease the computational complexity of the lower problem. Some of the above research ideas are already underway, but could not be shown here due to publication size restrictions.

References

1. Bożejko, W., Wodecki, M.: Parallel algorithm for some single machine scheduling problems. Zeszyty Naukowe. Automatyka, Politechnika Śląska **134**, 81–90 (2002)
2. Bragin, M.A., Luh, P.B., Yan, J.H., Yu, N., Stern, G.A.: Convergence of the surrogate Lagrangian relaxation method. J. Optim. Theory Appl. **164**, 173–201 (2015)
3. Butt, A.A., Collins, R.T.: Multi-target tracking by Lagrangian relaxation to min-cost network flow. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1846–1853 (2013)
4. Chen, H., Luh, P.B.: An alternative framework to Lagrangian relaxation approach for job shop scheduling. Eur. J. Oper. Res. **149**(3), 499–512 (2003)
5. Cui, H., Luo, X.: An improved Lagrangian relaxation approach to scheduling steelmaking-continuous casting process. Comput. Chem. Eng. **106**, 133–146 (2017)
6. Fisher, M.L.: A dual algorithm for the one-machine scheduling problem. Math. Program. **11**(1), 229–251 (1976)
7. Fisher, M.L.: The Lagrangian relaxation method for solving integer programming problems. Manage. Sci. **50**(12_suppl.), 1861–1871 (2004)
8. Fu, Y.M., Diabat, A.: A Lagrangian relaxation approach for solving the integrated quay crane assignment and scheduling problem. Appl. Math. Model. **39**(3–4), 1194–1201 (2015)
9. Gaudioso, M., Gorgone, E., Labbé, M., Rodríguez-Chía, A.M.: Lagrangian relaxation for SVM feature selection. Comput. Oper. Res. **87**, 137–145 (2017)
10. Gocgun, Y., Ghate, A.: Lagrangian relaxation and constraint generation for allocation and advanced scheduling. Comput. Oper. Res. **39**(10), 2323–2336 (2012)
11. Held, M., Wolfe, P., Crowder, H.P.: Validation of subgradient optimization. Math. Program. **6**, 62–88 (1974)
12. Imai, A., Nishimura, E., Current, J.: A Lagrangian relaxation-based heuristic for the vehicle routing with full container load. Eur. J. Oper. Res. **176**(1), 87–105 (2007)
13. Nishi, T., Hiranaka, Y., Inuiguchi, M.: Lagrangian relaxation with cut generation for hybrid flowshop scheduling problems to minimize the total weighted tardiness. Comput. Oper. Res. **37**(1), 189–198 (2010)
14. Nowak, M.P., Römisch, W.: Stochastic Lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. Ann. Oper. Res. **100**, 251–272 (2000)
15. ORlibrary. http://people.brunel.ac.uk/~mastjjb/jeb/info.html
16. Villarreal, F.J., Bulfin, R.L.: Scheduling a single machine to minimize the weighted number of tardy jobs. AIIE Trans. **15**(4), 337–343 (1983)
17. Wodecki, M.: A branch-and-bound parallel algorithm for single-machine total weighted tardiness problem. Int. J. Adv. Manuf. Technol. **37**(9), 996–1004 (2008)