Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 2730 – 2745
© TÜBİTAK
doi:10.3906/elk-1811-40

Modified self-adaptive local search algorithm for a biobjective permutation flow shop scheduling problem

Çiğdem ALABAŞ USLU[1,*], **Berna DENGİZ**[2], **Canan AĞLAN**[1], **İhsan SABUNCUOĞLU**[3]

[1]Department of Industrial Engineering, Faculty of Engineering, Marmara University, İstanbul, Turkey
[2]Department of Industrial Engineering, Faculty of Engineering, Başkent University, Ankara, Turkey
[3]Department of Industrial Engineering, Faculty of Engineering, Abdullah Gül University, Kayseri, Turkey

Received: 06.11.2018 • **Accepted/Published Online:** 05.05.2019 • **Final Version:** 26.07.2019

Abstract: Interest in multiobjective permutation flow shop scheduling (PFSS) has increased in the last decade to ensure effective resource utilization. This study presents a modified self-adaptive local search (MSALS) algorithm for the biobjective permutation flow shop scheduling problem where both makespan and total flow time objectives are minimized. Compared to existing sophisticated heuristic algorithms, MSALS is quite simple to apply to different biobjective PFSS instances without requiring effort or time for parameter tuning. Computational experiments showed that MSALS is either superior to current heuristics for Pareto sets or is incomparable due to other performance indicators of multiobjective problems.

Key words: Biobjective permutation flow shop, self-adaptive heuristic, parameter tuning

1. Introduction

Permutation flow shop scheduling (PFSS) is one of the most studied problems in operations research. The problem is to find the order of n jobs processed through a series of m machines, such that the processing order of the jobs is the same for all machines. Each job i has one operation on each machine j with a processing time $p_{ij} \geq 0$. Many optimization criteria have been proposed for the PFSS problem; for example, in [1] it was stated that the most common objective is minimization of makespan (C_{max}) followed by total flow time (TFT). On the other hand, interest in multiobjective PFSS problems has increased over the last decade due to the practical importance of ensuring effective resource utilization [2]. In multiobjective PFSS problems, several objectives are considered simultaneously because optimizing one criterion may cause deterioration of others. A multiobjective problem is specifically called a biobjective problem if it deals with optimization of only two conflicting objectives. The biobjective C_{max}-TFT has attracted substantial attention among all multiobjective approaches, since C_{max} minimization provides stable utilization of machine and manpower while TFT provides reduced work-in-process costs.

PFSS problems are NP-hard problems even for a single objective and that is why there are numerous state-of-the-art metaheuristic algorithms for the approximate solutions of the problem. Referring interested readers to the comprehensive reviews of [3] and [4], Section 2 covers the studies that presented metaheuristic algorithms for biobjective PFSS problems. The metaheuristic algorithms given in Section 2 are all managed by a set of parameters, which has a significant impact on the solution quality and/or computational time. Searching for an

*Correspondence: cigdem.uslu@marmara.edu.tr ideal parameter set, called parameter tuning or parameter optimization, is itself a difficult problem. Therefore, sophisticated metaheuristics, which generally have increasing numbers of parameters, suffer from complicated parameter tuning. The existing parameter tuning methods are classified as offline or online methods. Offline parameter tuning is implemented before the execution of the algorithm, whereas online methods, also known as parameter control techniques, are applied while running the algorithm.

Automatic algorithm configuration has increased interest in offline techniques, and it incorporates experimental design and statistical modeling techniques [5–9], racing algorithms [10–13], and metaoptimization approaches, which tune the parameters using any other heuristic [14–20].

Although many studies have focused on offline techniques, there are two main difficulties that remain: parameter values depend on the problem type and its various instances, and parameter effectiveness may vary at different moments in the search. Online or parameter control techniques overcome these difficulties by tuning parameters during the search process, updating dynamically or adaptively. There are many metaheuristic applications with various parameter control approaches, and readers are referred to the representative studies of [21–27], which mainly focus on adaptive parameters. On the other hand, these algorithms still include some parameters that need to be initialized. The work in [28] proposed a simple self-adaptive local search (SALS) and applied it to several single-objective problems including quadratic assignment, PFSS, classical vehicle routing, and topological optimization of backbone networks. The work in [29] also represented a detailed implementation of SALS for topological optimization problem. The only multiobjective application of SALS was provided in [30] for a biobjective vehicle routing problem. The SALS algorithm has a single parameter, called an acceptance parameter, which is updated using useful information (number of improved solutions and improvement rate of the initial solution cost) gathered throughout the search process. Trial solutions obtained during the search process are accepted or rejected depending on the current value of the acceptance parameter. It was shown that the tuning of the single parameter of SALS is independent of the problem type or its instances [28].

There are two main motivations of the present study: it is the first application of SALS to a biobjective PFSS problem considering C_{max} and TFT, and it modifies SALS (MSALS) by enhancing new acceptance rules. Although there are numerous studies in the literature of biobjective optimization of C_{max}-TFT, all of them suggest state-of-the-art heuristics, which are controlled by a set of well-tuned parameters. Parameter tuning itself is an optimization problem that necessitates deep knowledge of both the problem and the technique selected for tuning. Therefore, there is still room to investigate heuristic algorithms that are free of parameter tuning in the related literature. The present study contributes to the literature by introducing the MSALS algorithm, which is simple (from the perspective of parameter tuning) and efficient (from the perspective of obtaining larger sets of nondominated solutions). Thus, it is expected that MSALS will be easily adapted in the practice of PFSS problems with objectives of C_{max} and TFT. Experimental results also verify that MSALS tends to find a larger set of efficient solutions compared to SALS and it also surpasses most of the best state-of-the-art heuristics considering different evaluation criteria.

The present study continues with the explanation of the metaheuristics selected from the literature of multiobjective PFSS. Sections 3 and 4 present the basic SALS and MSALS algorithms, respectively. Section 5 provides the comparison of the proposed MSALS approach with SALS and also with the state-of-the-art heuristics, which performs superiorly in the instances of [31]. Section 6 summarizes and concludes the paper, and discusses future research directions.

2. Related studies

In [4], Minella et al. evaluated 23 heuristic algorithms for multiobjective PFSS problems and showed that the multiobjective simulated annealing (MOSA) of [32], multiobjective genetic algorithm local search (MOGALS) of [33], and multiobjective tabu search (MOTS) of [34] exhibit superior performance compared to the other evaluated algorithms. Later, the authors of [35] proposed the multiobjective iterated greedy search (MOIGS) algorithm using a greedy mechanism, which has better performance than MOSA in terms of nondominated solutions.

The authors of [36] proposed the multiobjective ant colony algorithm (MOACA) with 20 variants by changing parameter settings and the local search scheme sequence. MOACA variants were compared with the algorithms in [4] and MOIGS, showing that most nondominated solutions in the Pareto sets were yielded by MOACA.

The authors of [37] proposed the two-phase Pareto local search (TP+PLS) algorithm, combining two-phase local search and Pareto local search paradigms. Two-phase local search aggregates multiple objective functions into a single objective, whereas the Pareto local search investigates nondominated solutions for the considered initial set. Iterated greedy algorithms were integrated into the two-phase local search framework for the multiobjective case as an initial step, with the Pareto local search algorithm implemented in the next step to explore the search space. The two algorithms were combined to solve the biobjective PFSS problem for the selected biobjectives among the pairs of C_{max}, TFT, and weighted/unweighted total tardiness. It was shown in [37] that TP+PLS contributed a high percentage of nondominated solutions to the net fronts.

The work in [1] proposed the restarted iterated Pareto greedy algorithm (RIPG), incorporating a tailored local search mechanism and an effective initialization procedure. RIPG showed similar performance to state-of-the-art algorithms for small problems, but significantly superior performance for medium or large problems.

The relatively recent study in [38] presented biobjective multistart simulated annealing (BMSA), which combines simulated annealing and a hill climbing strategy to achieve convergence of search and escape from local optima. BMSA performance was compared with 8 algorithms, excluding TP+PLS and RIPG, and it was shown that BMSA contributes to 64% of the Pareto sets of the instances of [31].

The studies mentioned above presented the best state-of-the-art algorithms tested on the benchmark instances of [31]. Therefore, the comparative study presented in Section 5 includes only these algorithms to provide a comparison on the same basis. Although there are several recent studies in the literature that concentrated on C_{max} and TFT objectives, such as [39], [40], and [41], they considered neither the same performance indicators of this study nor the same problem instances. Additionally, these studies did not present the Pareto set to perform comparisons.

3. Basic structure of self-adaptive local search algorithm

The SALS algorithm starts with a solution, **X**, and searches the neighborhood of the current solution iteratively. At each iteration a neighbor solution, **X'**, is selected randomly from the neighborhood of the current solution, N(**X**), and accepted as the new current solution if **X'** satisfies the following acceptance rule, where $f(\mathbf{X})$ is the objective function to be minimized:

If $f(\mathbf{X'}) \leqslant \Theta f(\mathbf{X})$ then $\mathbf{X} \longleftarrow \mathbf{X'}$.

According to the given acceptance rule, a single parameter of SALS, Θ, determines the vicinity of the current solution. Θ relies on two criteria: the quality of the best solution and the number of improved solutions obtained throughout the search process. The work in [28] introduced α_1 and α_2, by Eqs. (1) and (2), to measure the quality of the best solution and the number of the improved solutions, respectively. In Eq. (1), $\mathbf{X}_b{}^{(\mathbf{i})}$ is the best solution found until iteration i and $\mathbf{X}_z$ is the initial solution. f($\mathbf{X}$) is assumed to be positive for the whole solution space. C(L $^{(i)}$) given in Eq. (2) is a counter, which is updated whenever a new best solution, $\mathbf{X}_b{}^{(\mathbf{i})}$, is found. The search process is kept in the neighborhood of the current solution, N($\mathbf{X}$), until the acceptance rule is assured. If the total number of rejected neighbors reaches the neighborhood size, $|\, N(\mathbf{X})\,|$, Θ is increased by an amount of $\alpha_1\,\alpha_2$ only for the current neighborhood. The algorithm proceeds to the next iteration whenever a neighbor solution, **X'**, is accepted. Θ is readjusted using α_1 and α_2 at the beginning of each iteration as given in Eq. (3). The basic steps of SALS are given in Figure 1. As shown in Figure 1, Θ does not require a predefined initial value since it is automatically equal to 2. As the search process proceeds, α_1 and α_2 make Θ smaller. If the algorithm is allowed to run enough, Θ approaches 1 and forces the search process to find better solutions. A detailed analysis of Θ can be found in [28].

$$\alpha_1 = \frac{f(\mathbf{X}_b{}^{(\mathrm{i})})}{f(\mathbf{X}_z)} \tag{1}$$

$$\alpha_2 = \frac{C(L^{(\mathrm{i})})}{i} \tag{2}$$

$$\Theta = 1 + \alpha_1\alpha_2 \tag{3}$$

```
$i \leftarrow 1$, $C(L^{(i)}) \leftarrow 1$
Randomly generate initial solution $\mathbf{X}_z$
$\mathbf{X} \leftarrow \mathbf{X}_z$; $\mathbf{X}_b^{(i)} \leftarrow \mathbf{X}_z$
Repeat
  $\alpha_1 \leftarrow \frac{f(\mathbf{X}_b^{(i)})}{f(\mathbf{X}_z)}$; $\alpha_2 \leftarrow \frac{C(L^{(i)})}{i}$
  $\Theta \leftarrow 1 + \alpha 1 \alpha 2$
  $i \leftarrow i + 1$; $r \leftarrow 0$
  Repeat
    Select a neighbor solution $\mathbf{X'}$ randomly from the $N'(\mathbf{X})$
    $r \leftarrow r + 1$
    if $r = |N(\mathbf{X})|$ then $\Theta \leftarrow \Theta + \alpha_1\alpha_2$
  Until $f(\mathbf{X'}) \le \Theta f(\mathbf{X})$
  If $f(\mathbf{X'}) < f(\mathbf{X}_b^{(i)})$ then $C(L^{(i)}) \leftarrow C(L^{(i)}) + 1$,
   $\mathbf{x}_b^{(i)} \leftarrow \mathbf{x}'$
  $\mathbf{X} \leftarrow \mathbf{X'}$
Until $\Theta \rightarrow 1$
```

Figure 1. Steps of SALS.

4. MSALS for biobjective PFSS problems

MSALS, for biobjective PFSS problems, uses the permutation representation of a solution (i.e. a sequence of the jobs), $\mathbf{X} = [\, x_1, ...x_s, ...x_n]$, where x_s corresponds to a job in the sth order.

The neighbor solutions are created using three types of moving mechanisms: adjacent-swap, exchange, and insertion. The adjacent-swap mechanism interchanges x_s with either x_{s-1} or x_{s+1} where s+1 $\leq$ n and s-1$\geq$ 1. The exchange mechanism interchanges x_s with x_p, where p-s > 1, and the insertion mechanism inserts x_s between x_p and $x_{\mathrm{p+1}}$ where p+1$\leq$ n.

The same moving mechanisms are also utilized by SALS when the permutation representation is adapted [28, 30, 42]. MSALS selects one of the moving types randomly and then generates a random neighbor using the selected moving mechanism at each iteration, allowing repetitions of the neighbors. If the candidate neighbor solution, **X'**, meets the acceptance rule defined below, it is recorded as the new current solution. MSALS manipulates both C$_{\mathrm{max}}$ and TFT objectives simultaneously for the neighbor solutions using the following three acceptance rules in turn:

Rule$_1$: if $\{C_{\mathrm{max}}(\mathbf{X'}) \leq \Theta^{\mathrm{Cmax}} C_{\mathrm{max}}(\mathbf{X})\}$ and $\{\mathrm{TFT}(\mathbf{X'}) \leq \Theta^{\mathrm{TFT}} \mathrm{TFT}(\mathbf{X})\}$ then $\mathbf{X} \longleftarrow \mathbf{X'}$

Rule$_2$: if $\{C_{\mathrm{max}}(\mathbf{X'}) \leq \Theta^{\mathrm{Cmax}} BestC^{(\mathrm{i})}{}_{\mathrm{max}}\}$ and $\{\mathrm{TFT}(\mathbf{X'}) \leq \Theta^{\mathrm{TFT}} \mathrm{TFT}(\mathbf{X})\}$ then $\mathbf{X} \longleftarrow \mathbf{X'}$

Rule$_3$: if $\{C_{\mathrm{max}}(\mathbf{X'}) \leq \Theta^{\mathrm{Cmax}} C_{\mathrm{max}}(\mathbf{X})\}$ and $\{\mathrm{TFT}(\mathbf{X'}) \leq \Theta^{\mathrm{TFT}} BestTFT^{(\mathrm{i})}\}$ then $\mathbf{X} \longleftarrow \mathbf{X'}$

Parameters Θ^{Cmax} and Θ^{TFT} are introduced for C$_{\mathrm{max}}$ and TFT objectives, respectively, as given in Eqs. (8) and (9). According to the first acceptance rule, a neighbor solution is accepted as a new current solution if its C$_{\mathrm{max}}$ is better than Θ^{Cmax} times the current solution's C$_{\mathrm{max}}$ and its TFT is better than Θ^{TFT} times the current solution's TFT. The second and third rules, structurally, are similar to the first rule. However, the second rule takes account of the best C$_{\mathrm{max}}$ and the last rule uses the best TFT found until the current iteration instead of C$_{\mathrm{max}}$ and TFT values of the current solution, respectively. MSALS is run using each of the rules for a prespecified number of replications. Whenever a neighbor solution is accepted based on the given acceptance rule, the algorithm proceeds to the next iteration. If the total number of rejected neighbors exceeds the neighborhood size then Θ^{Cmax} and Θ^{TFT} are increased by ${\alpha_1}^{\mathrm{Cmax}}$ ${\alpha_2}^{\mathrm{Cmax}}$ and ${\alpha_1}^{\mathrm{TFT}}$ ${\alpha_2}^{\mathrm{TFT}}$, respectively. Once the algorithm proceeds to the next iteration, Θ^{Cmax} and Θ^{TFT} are recomputed by Eqs. (8) and (9) using the current levels of ${\alpha_1}^{\mathrm{Cmax}}$, ${\alpha_2}^{\mathrm{Cmax}}$ and ${\alpha_1}^{\mathrm{TFT}}$, ${\alpha_2}^{\mathrm{TFT}}$, respectively. The ${\alpha_1}^{\mathrm{Cmax}}$, ${\alpha_2}^{\mathrm{Cmax}}$, ${\alpha_1}^{\mathrm{TFT}}$, and ${\alpha_2}^{\mathrm{TFT}}$ are computed as given in Eqs. (4)–(7) as in SALS. In these equations, ${C_{\mathrm{Cmax}}}^{(\mathrm{i})}$ and ${C_{\mathrm{TFT}}}^{(\mathrm{i})}$ counters are updated whenever C$_{\mathrm{max}}$ of **X'** is better than the best C$_{\mathrm{max}}$, ${BestC_{Cmax}}^{(i)}$, found until iteration i and TFT of **X'** is better than the best TFT, $BestTFT^{(i)}$, found until iteration i.

$$ {\alpha_1}^{\mathrm{Cmax}} = \frac{BestC_{\mathrm{max}}{}^{(\mathrm{i})}}{InitC_{\mathrm{max}}} \tag{4} $$

$$ {\alpha_2}^{\mathrm{Cmax}} = \frac{C_{\mathrm{Cmax}}{}^{(\mathrm{i})}}{i} \tag{5} $$

$$ {\alpha_1}^{\mathrm{TFT}} = \frac{BestTFT^{(\mathrm{i})}}{InitTFT} \tag{6} $$

$$ {\alpha_2}^{\mathrm{TFT}} = \frac{C_{\mathrm{TFT}}{}^{(\mathrm{i})}}{i} \tag{7} $$

$$ \Theta^{\mathrm{Cmax}} = 1 + {\alpha_1}^{\mathrm{Cmax}} + {\alpha_2}^{\mathrm{Cmax}} \tag{8} $$

$$ \Theta^{\mathrm{TFT}} = 1 + {\alpha_1}^{\mathrm{TFT}} + {\alpha_2}^{\mathrm{TFT}} \tag{9} $$

Finally, the basic steps of MSALS are given in Figure 2. During the initial iterations of the algorithm the search is almost random, since both Θ^{Cmax} and Θ^{TFT} are initially equal to 2. It is expected that as Θ^{Cmax} and Θ^{TFT} approach 1, the search converges to the set of high quality solutions.

```
Rule$_1$ ← if {$C_{\max}(\mathbf{X'}) \le \theta^{Cmax} C_{\max}(\mathbf{X})$}and{ $TFT(\mathbf{X'}) \le \theta^{TFT}\, TFT(\mathbf{X})$} then accept$\mathbf{X'}$=true;
Rule$_2$ ← if {$C_{\max}(\mathbf{X'}) \le \theta^{Cmax} C_{\max}(\mathbf{x}_b^{(i)})$}and{ $TFT(\mathbf{X'}) \le \theta^{TFT}\, TFT(\mathbf{X})$} then accept$\mathbf{X'}$=true;
Rule$_3$ ← if {$C_{\max}(\mathbf{X'}) \le \theta^{Cmax} C_{\max}(\mathbf{X})$)}and{ $TFT(\mathbf{X'}) \le \theta^{TFT}\, TFT(\mathbf{x}_b^{(i)})$} then accept$\mathbf{X'}$=true;
$k \leftarrow 0$;
Repeat
  $k \leftarrow k+1$; AcceptanceRule ← Rule$_k$;
  $i \leftarrow 1$; $C_{C_{\max}}^{(i)} \leftarrow 1$; $C_{TFT}^{(i)} \leftarrow 1$;
  Randomly generate an initial solution $\mathbf{X}$;
  $InitC_{\max} \leftarrow C_{\max}(\mathbf{X})$; $InitTFT \leftarrow TFT(\mathbf{X})$; $BestC_{\max}^{(i)} \leftarrow C_{\max}(\mathbf{X})$; $BestTFT^{(i)} \leftarrow TFT(\mathbf{X})$
  $\alpha_1^{Cmax} \leftarrow \frac{BestC_{\max}^{(i)}}{InitC_{\max}}$; $\alpha_2^{Cmax} \leftarrow \frac{C_{C_{\max}}^{(i)}}{i}$; $\theta^{Cmax} \leftarrow 1+\alpha_1^{Cmax}\alpha_2^{Cmax}$;
  $\alpha_1^{TFT} \leftarrow \frac{BestTFT^{(i)}}{InitTFT}$; $\alpha_2^{TFT} \leftarrow \frac{C_{TFT}^{(i)}}{i}$; $\theta^{TFT} \leftarrow 1+\alpha_1^{TFT}\alpha_2^{TFT}$;
        Repeat
          $i \leftarrow i+1$; $r \leftarrow 0$;
          Repeat
            Select a neighbor solution $\mathbf{X'}$ randomly from the $N'(\mathbf{X})$;
            $r \leftarrow r+1$;
            if $r = |N(\mathbf{X})|$ then $\theta^{Cmax} \leftarrow \theta^{Cmax} + \alpha_1^{Cmax}\alpha_2^{Cmax}$; $\theta^{TFT} \leftarrow \theta^{TFT} + \alpha_1^{TFT}\alpha_2^{TFT}$;
            AcceptanceRule;
          Until accept$\mathbf{X'}$=true;
          If $C_{\max}(\mathbf{X'}) < BestC_{\max}^{(i)}$ then
            $C_{Cmax}^{(i)} \leftarrow C_{Cmax}^{(i-1)} + 1$, $BestC_{\max}^{(i)} \leftarrow C_{\max}(\mathbf{X'})$,
            Record $\mathbf{X'}$, $C_{\max}(\mathbf{X'})$, $TFT(\mathbf{X'})$ in ElitList;
          If $TFT(\mathbf{X'}) < BestTFT^{(i)}$ then
            $C_{TFT}^{(i)} \leftarrow C_{TFT}^{(i-1)} + 1$, $BestTFT^{(i)} \leftarrow TFT(\mathbf{X'})$,
            Record $\mathbf{X'}$, $C_{\max}(\mathbf{X'})$, $TFT(\mathbf{X'})$ in ElitList;
          $\alpha_1^{Cmax} \leftarrow \frac{BestC_{\max}^{(i)}}{InitC_{\max}}$; $\alpha_2^{Cmax} \leftarrow \frac{C_{C_{\max}}^{(i)}}{i}$; $\theta^{Cmax} \leftarrow 1+\alpha_1^{Cmax}\alpha_2^{Cmax}$;
          $\alpha_1^{TFT} \leftarrow \frac{BestTFT^{(i)}}{InitTFT}$; $\alpha_2^{TFT} \leftarrow \frac{C_{TFT}^{(i)}}{i}$; $\theta^{TFT} \leftarrow 1+\alpha_1^{TFT}\alpha_2^{TFT}$;
          $\mathbf{X} \leftarrow \mathbf{X'}$;
          $C_{\max}(\mathbf{X}) \leftarrow C_{\max}(\mathbf{X'})$;
          $TFT(\mathbf{X}) \leftarrow TFT(\mathbf{X'})$;
        Until ($\theta^{Cmax} \rightarrow 1$) or ($\theta^{TFT} \rightarrow 1$) or (Any given termination criterion is met);
Until $k$ =3;
Filter nondominated solutions from ElitList;
```

Figure 2. Steps of MSALS.

5. Computational study

Two experimental studies were conducted. Section 5.1 investigates the proposed MSALS algorithm's acceptance rules, and Section 5.2 compares the proposed MSALS outcomes with a reference set of benchmark instances and several state-of-the-art heuristics of the biobjective PFSS. All experiments were performed on the well-known and widely utilized [31] benchmarks. The 110 Taillard instances were classified into 11 groups: nine combinations of $\{20, 50, 100\}$ jobs and $\{5, 10, 20\}$ machines, and two combinations of 200 jobs and $\{10, 20\}$ machines, with 10 instances in each group.

5.1. Analysis of the proposed acceptance rules

The application of the SALS algorithm to biobjective vehicle routing problems, called SALS-AU, was proposed in [30]. The difference between MSALS and SALS-AU is the acceptance rules, explained as follows: let Θ_1 and Θ_2 represent the acceptance parameters for the first and second objectives (in this study Θ_1 is Θ^{Cmax} and Θ_2 is Θ^{TFT}), and $f_1(\mathbf{X})$ and $f_2(\mathbf{X})$ are the first and second objective functions of solution $\mathbf{X}$, respectively ($f_1(\mathbf{X})$ is $\mathrm{C}_{\max}(\mathbf{X})$ and $f_2(\mathbf{X})$ is $\mathrm{TFT}(\mathbf{X})$. $f_1({x_b}^{(\mathrm{i})})$ and $f_2({x_b}^{(\mathrm{i})})$ are used to indicate the best value until iteration i of the first and second objective functions, respectively ($f_1({x_b}^{(\mathrm{i})})$ is $BestC_{\max}{}^{(\mathrm{i})}$ and ($f_2({x_b}^{(\mathrm{i})})$ is $BestTFT^{(\mathrm{i})}$). While SALS-AU uses the single acceptance rule as "($f_1\mathbf{X'} \leq \Theta_1 f_1(\mathbf{X})$) and ($f_2\mathbf{X'} \leq \Theta_2 f_2(\mathbf{X})$)", MSALS uses two additional acceptance rules given in Table 1.

Table 1. Acceptance rules of MSALS and SALS-AU.

MSALS	SALS-AU
$\{f_1(\mathbf{X'}) \leq \Theta_1 f_1(\mathbf{X})\}$ and $\{f_2(\mathbf{X'}) \leq \Theta f_2(\mathbf{X})\}$	$\{f_1(\mathbf{X'}) \leq \Theta_1 f_1(\mathbf{X})\}$ and $\{f_2(\mathbf{X'}) \leq \Theta f_2(\mathbf{X})\}$
$\{f_1(\mathbf{X'}) \leq \Theta_1 f_1({x_b}^{(\mathrm{i})})\}$ and $\{f_2(\mathbf{X'}) \leq \Theta f_2(\mathbf{X})\}$	
$\{f_1(\mathbf{X'}) \leq \Theta_1 f_1(\mathbf{X})\}$ and $\{f_2(\mathbf{X'}) \leq \Theta_2 f_2({x_b}^{(\mathrm{i})})\}$	

MSALS needs to be run sequentially for each of the three acceptance rules as explained in Section 4. An equal number of runs (20 runs) is set for the MSALS and SALS-AU algorithms. Runs were performed on the same PC (Intel core 2 duo CPU 2.93 GHz) for each instance of [31]), but for instances with 200 jobs. Each run of the two algorithms is terminated when a predetermined number of evaluated solutions is met. In this comparative study, we aim to show the contribution of the additional two acceptance rules of MSALS to the nondominated solution set. The experiments with MSALS also showed that its run time is short enough to apply even large sizes of the problem (see Table 6). As an illustrative example, Figure 3 represents efficient solutions obtained by applying each acceptance rule separately and nondominated solutions filtered from these efficient solutions (notated as *all rules* in the figure) for an instance with size 100 × 20. Each rule clearly covers a different portion of the Pareto front, since rules 2 and 3 substantially force the search process to improve $\mathrm{C}_{\max}$ and TFT, respectively, while rule 1 tempers these two rules.

The utilization of all three acceptance rules results in larger sets of efficient solutions as shown in Table 2. Apparently, the three acceptance rules of MSALS cause a substantial improvement in the number of efficient solutions compared to SALS-AU, in which only the first rule is adopted. It is also notable that as the problem size increases, the number of efficient solutions by MSALS tends to increase as shown in the right-most column of Table 2.

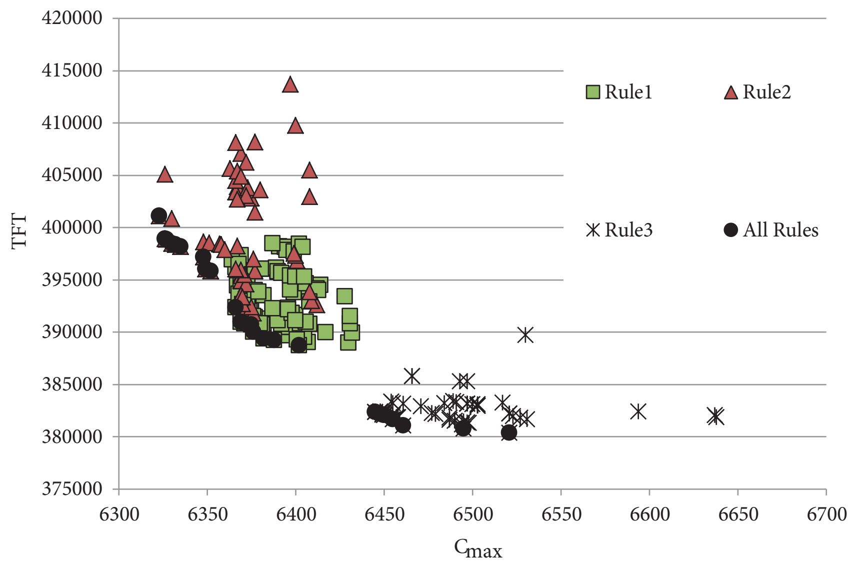


Figure 3. Efficient solutions obtained by each rule and nondominated solutions by the three rules for instance DD_Ta084 with size 100 × 20.

Table 2. Number of efficient solutions obtained by MSALS and SALS-AU.

		PR1	PR2	PR3	PR4	PR5	PR6	PR7	PR8	PR9	PR10	Total	Improvement %
20 × 5	SALS-AU	1	6	9	6	9	10	6	12	9	7	75	37
	MSALS	3	6	9	11	13	13	12	13	13	10	103	
20 × 10	SALS-AU	6	13	7	12	12	17	11	6	8	8	100	45
	MSALS	17	16	9	20	15	23	13	11	13	8	145	
20 × 20	SALS-AU	18	13	13	14	9	9	8	11	10	9	114	80
	MSALS	27	22	28	16	16	17	15	28	17	19	205	
50 × 5	SALS-AU	4	4	6	8	2	4	4	1	4	3	40	70
	MSALS	5	5	11	7	6	8	6	8	6	6	68	
50 × 10	SALS-AU	6	7	15	5	10	8	7	9	8	9	84	101
	MSALS	14	16	27	13	15	15	12	20	19	18	169	
50 × 20	SALS-AU	11	11	11	5	8	16	12	8	10	7	99	93
	MSALS	22	20	19	16	17	24	16	18	21	18	191	
100 × 5	SALS-AU	3	4	5	3	2	2	5	5	9	2	40	80
	MSALS	6	5	10	6	7	5	9	11	8	5	72	
100 × 10	SALS-AU	6	5	2	8	12	4	6	11	2	3	59	92
	MSALS	12	8	8	10	19	16	12	14	8	6	113	
100 × 20	SALS-AU	6	9	6	16	8	8	16	13	10	4	96	152
	MSALS	24	25	26	32	16	22	25	28	25	19	242	

5.2. Comparison of MSALS with the literature

A set of efficient solutions obtained from a multiobjective method is an approximation to the optimal Pareto front. One common method to compare the set of solutions of different algorithms is to show Pareto dominance relations between the sets. Namely, a solution set X dominates set Y if and only if every solution in set Y is dominated by at least one solution in set X. To compare the MSALS algorithm with the existing heuristic algorithms in the literature in this respect, we use the performance metrics given below considering the notations of [38]:

A: The number of efficient solutions found by an algorithm.

B: The number of nondominated solutions found by an algorithm in the Pareto front.

C(X,Y): The ratio of the number of dominated solutions of algorithm X by algorithm Y to the number of efficient solutions of algorithm X.

In this study, the hypervolume indicator, I_H, and its unary version, called the epsilon indicator, I_ϵ, which are commonly used in the multiobjective literature (see [43]), are utilized. The I_H indicator measures the hypervolume of the objective space dominated by a given Pareto set. In biobjective problems, I_H measures the area covered by a given Pareto set with respect to a reference point. Therefore, having a higher hypervolume means a better frontier. By considering the related literature, the reference point is obtained by increasing the worst objective value by 20%. Since the objective values are normalized between [0,1], the largest I_H (i.e. the maximum area) is the multiplication of the reference points, r_1 and r_2 ($1.2 \times 1.2 = 1.44$), for biobjectives. The I_H indicator is computed by Eq. (10) for the biobjectives, where $s \in S$ represents the solutions of an approximation to the Pareto front and $\overline{O}_i(\mathrm{i})$ is the ith normalized objective value of solution s.

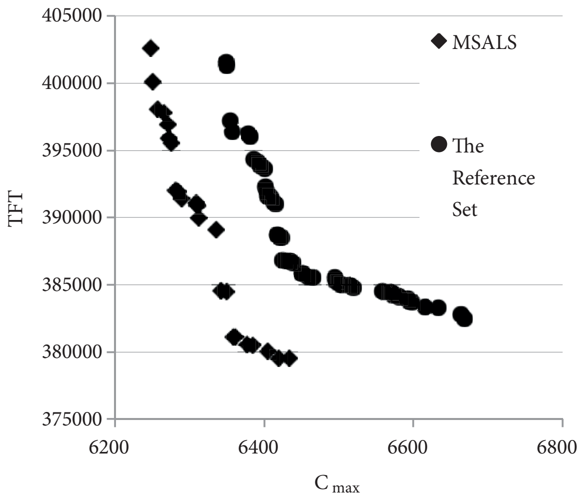
$$I_H = \cup_{s \in S} [r_1, \overline{O}_1(s)] \times [r_2, \overline{O}_2(s)] \tag{10}$$

The I_ϵ indicator gives the minimum distance between a given Pareto front and the optimum Pareto front (or a reference set); hence, lower values of I_ϵ show a better frontier. I_ϵ is computed for the biobjectives as given by Eq. ((11), where R is the reference set and S is an approximation of the Pareto front.

$$I_\epsilon = max \left\{ \frac{min_{s \in S}(O_1(s))}{min_{s \in R}(O_1(s))}, \frac{max_{s \in S}(O_1(s))}{max_{s \in R}(O_1(s))}, \frac{min_{s \in S}(O_2(s))}{min_{s \in R}(O_2(s))}, \frac{max_{s \in S}(O_2(s))}{max_{s \in R}(O_2(s))} \right\} \tag{11}$$

Minella et al. [4] presented a more detailed review and the evaluation of 23 heuristic algorithms for three different bicriteria flow shop scheduling problems. The preliminary results showed that MSALS outperforms the reference sets of Taillard's instances in [4]. Figure 4 shows illustrative examples for comparison of MSALS with the reference sets on instances sized 100 × 20 and 200 × 20. It is observed that MSALS is able to dominate the reference sets in [4] even though it finds a smaller set of efficient solutions.

According to the review of [4], MOSA [32] was the best performing algorithm for the $\mathrm{C}_{\mathrm{max}}$-TFT biobjective among the investigated heuristic algorithms. Later, MOIGS [35] outperformed MOSA, and afterwards, MOACA [36] gave better results than MOIGS. The authors of [36] also reported the best approximations to the Pareto sets of the instances of [31] to date. In this study, the reported Pareto fronts by [36] were notated as P_{RZ}. P_{RZ} is made up of the solutions provided by MOACA, MOSA, and MOIGS. Besides these three heuristic algorithms, RIGP [1], TP+PLS [37], and BMSA [38] are other state-of-the-art algorithms proposed for the biobjective PFSS problem. The performance of MSALS was compared with the solutions represented in P_{RZ}, and the results of BMSA, TP+PLS, and RIGP. It should be noted that the Pareto fronts provided by BMSA are from the study of [38], the Pareto fronts of each experiment of TP+PLS are from the website http://iridia.ulb.ac.be/ jdubois/pfsp_refsets.htm, and RIGP is reexperimented on the benchmark instances using an executable file provided by the corresponding author of [1]. The termination criterion of BMSA is the number of solutions searched and it is stated as a function of n and m. MSALS is allowed to run up to a maximum number of solutions searched, too. The maximum number is the same as that of BMSA given by the



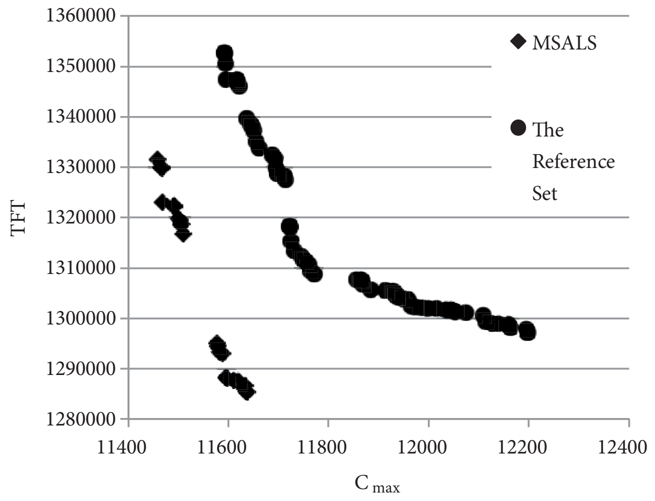


Figure 4. Efficient solution sets of MSALS against the reference sets of [4] for instances DD_Ta082 (100 × 20) on the left and DD_Ta103 (200 × 20) on the right.

first set of parameters in [38]. On the other hand, TP+PLS and RIGP are terminated when a predetermined time limit, which is 0.1 nm, is reached. MSALS was run 20 times on a PC with an Intel core 2 duo CPU, 2.93 GHz (the same experiments used to analyze the proposed acceptance rules).

To avoid confusion in the comparative study, an approximation to the Pareto set provided by an algorithm is called a set of efficient solutions while the set of nondominated solutions contributed by all algorithms is called the Pareto front. The number of efficient solutions, A, provided by RIGP, BMSA, TP+PLS, and MSALS and in the set P_{RZ} is averaged over ten instances of each size of instances [31] as given in Table 3. The number of nondominated solutions, B, in the Pareto front contributed by the four algorithms and P_{RZ} is also given in Table 3 in terms of averages over ten instances per size. Additionally, the B/A performance metric shows the ability of an algorithm to have more nondominated solutions in its set of efficient solutions. Results in Table 3 show that the TP+PLS algorithm generates, on average, the largest size of efficient solutions compared to other algorithms. RIGP also outperforms BMSA, MSALS, and the solutions in set P_{RZ} in terms of A, while BMSA and MSALS have similar performances. The average of nondominated solutions, B, contributed to the Pareto sets is best for TP+PLS. RIGP and MSALS contribute to the Pareto sets showing close performances, surpassing BMSA and the solutions of P_{RZ}. On the other hand, MSALS has the largest B/A ratio, which indicates most of the efficient solutions generated by MSALS also contribute to the Pareto sets. However, the performance of TP+PLS is close to MSALS in terms of B/A. RIGP and BMSA yield the same B/A ratios on average. Table 4 gives pairwise comparisons of MSALS with the three algorithms and P_{RZ} in terms of coverage criterion C. These results reveal that MSALS dominates nearly half of the efficient solutions of the other algorithms but TP+PLS. BMSA and set P_{RZ} are able to dominate only a small fraction of the efficient solutions of MSALS. However, TP+PLS and RIGP dominate 42% and 33% of the efficient solutions provided by MSALS, respectively. Finally, Table 5 gives the total number of nondominated solutions contributed by the algorithms to the Pareto fronts of Taillard instances (since solutions of the instances with size 200 × 10 and 200 × 20 are not available in the associated studies, we use NA in Tables 3–5). Clearly the largest contribution to the frontiers is provided by TP+PLS, while MSALS and RIGP have close performances. The complete set of the new Pareto fronts consolidated for the 110 instances of [31] is given at http://mimoza.marmara.edu.tr/ cigdem.uslu/Results_Taillard_Instances_Cmax_TFT.pdf to serve future researchers.

Table 3. Comparison of MSALS, BMSA, PRZ, TP+PLS, and RIGP according to pairwise comparison measures A and B.

	BMSA			P_{RZ}			MSALS			TP+PLS			RIGP		
n × m	A[1]	B[2]	B/A	A	B	B/A	A	B	B/A	A	B	B/A	A	B	B/A
20 × 5	13.4	12.7	0.95	10.3	9.7	0.94	10.3	5.5	0.53	14.6	13.2	0.90	16.1	15.7	0.98
20 × 10	17.5	22.1	0.73	18.4	13	0.71	14.5	9.3	0.64	20.7	13.6	0.66	22.1	19.2	0.87
20 × 20	18.7	15.7	0.84	26.7	20.7	0.78	20.5	14.6	0.71	28	22.2	0.79	30.4	28.2	0.93
50 × 5	8.4	0.5	0.06	7.1	0.2	0.03	6.8	7	1.03	20.1	7.1	0.35	19.5	0.7	0.04
50 × 10	18	0.6	0.03	10.8	0	0	16.9	6.6	0.39	47.3	35.2	0.74	37.3	3	0.08
50 × 20	17.4	0	0	12.9	0	0	19.1	4.3	0.23	53.3	41.2	0.77	37.1	3.6	0.10
100 × 5	8.8	1.6	0.18	6.9	0.4	0.06	7.2	3.4	0.47	21.8	6.6	0.30	20.6	0.5	0.02
100 × 10	14.5	1.8	0.12	15.1	0.6	0.04	11.3	5.7	0.5	46.8	26.4	0.56	40.1	3.8	0.09
100 × 20	26.2	0.4	0.02	7.8	0	0	24.2	12.7	0.52	78.8	36.2	0.46	52.3	14.2	0.27
200 × 10	NA	NA	NA	NA	NA	NA	11.1	5.7	0.51	39.5	12.3	0.31	32.9	6.5	0.20
200 × 20	NA	NA	NA	NA	NA	NA	24.4	23	0.94	72	3.4	0.05	60.3	6.2	0.10
Average	15.88	5.11	0.33	12.89	4.96	0.28	15.12	8.89	0.59	40.26	19.76	0.54	33.52	9.24	0.33

Table 4. Number of efficient solutions obtained by MSALS and SALS-AU.

	C(MSALS, BMSA)	C(MSALS, P_{RZ})	C(MSALS, TP+PLS)	C(MSALS, RIGP)	C(BMSA, MSALS)	C(P_{RZ}, SALS)	C(TP+PLS, MSALS)	C(RIGP, MSALS)
20 × 5	0.02	0.01	0.03	0.00	0.35	0.3	0.38	0.45
20 × 10	0.24	0.2	0.20	0.13	0.25	0.19	0.23	0.34
20 × 20	0.07	0.15	0.13	0.05	0.18	0.2	0.24	0.27
50 × 5	0.46	0.55	0.46	0.79	0.24	0.09	0.31	0.09
50 × 10	0.71	0.61	0.15	0.36	0.2	0.14	0.72	0.46
50 × 20	0.74	0.63	0.13	0.26	0.12	0.05	0.81	0.46
100 × 5	0.56	0.48	0.56	0.65	0.22	0.21	0.58	0.38
100 × 10	0.7	0.65	0.39	0.69	0.18	0.04	0.50	0.27
100 × 20	0.86	0.96	0.37	0.37	0.03	0	0.48	0.48
200 × 10	NA	NA	0.51	0.66	NA	NA	0.39	0.36
200 × 20	NA	NA	0.92	0.90	NA	NA	0.02	0.05
Average	0.49	0.47	0.35	0.44	0.19	0.13	0.42	0.33

Since the run time of the algorithms depends on the hardware and software available, and also programming skills, we report only the run time requirement of MSALS in Table 6 to give an idea about its computational performance. Table 6 shows that MSALS has enough efficiency to apply large-sized instances of the problem. Also, it should be noted that all the considered algorithms except MSALS require fine-tuning of their generic parameters in addition to the run time requirement on the instances. Five parameters of BMSA, 6 parameters of TP+PLS, and 6 parameters of RIGP are determined by the authors either through extensive preexperiments or an experimental design scheme, using substantial time and effort. The utilized self-tuning

Table 5. Total contribution to the Pareto fronts by MSALS, BMSA, PRZ, TP+PLS, and RIGP.

n × m	Number of nondominated solutions in the Pareto front	BMSA	P_{RZ}	MSALS	TP+PLS	RIGP
20 × 5	165	127	97	55	132	157
20 × 10	213	127	130	93	136	192
20 × 20	307	157	207	146	222	282
50 × 5	153	5	2	70	71	7
50 × 10	453	6	0	66	352	30
50 × 20	493	0	0	43	412	36
100 × 5	130	16	4	34	66	5
100 × 10	389	18	6	57	264	38
100 × 20	635	4	0	127	362	142
200 × 10	262[a]	NA	NA	61	123	65
200 × 20	326	NA	NA	230	34	62
Total number of contributed solutions	3526	460	446	982	2174	1016

[a]17 solutions out of 262 are reported in [4].

property allows MSALS to be applicable to different instances without any initialization or tuning of parameters and this is an apparent advantage of MSALS compared with the other algorithms.

Table 6. Average run time performances of MSALS.

n × m	MSALS *CPU*[3] (s)
20 × 5	1.98
20 × 10	4.02
20 × 20	7.02
50 × 5	11.49
50 × 10	19.98
50 × 20	43.98
100 × 5	40.98
100 × 10	77.49
100 × 20	160.02
Average	40.77

According to the results in Tables 3–5, TP+PLS shows better performance than the other algorithms in generating efficient solutions and contributing to the Pareto sets with more nondominated solutions. MSALS has the best ratio of nondominated solutions to the number of efficient solutions. MSALS and RIGP show similar performances with respect to the average of nondominated solutions while MSALS and BMSA show similar performances for the number of efficient solutions on average. Coverage of MSALS over each algorithm is greater than the coverage of each algorithm over MSALS, except in the comparison of MSALS with TP+PLS. Consequently, TP+PLS, RIGP, and MSALS can be stated as the best three algorithms for C_{max}-TFT biobjectives of PFSS problems. Additionally, hypervolume and epsilon indicators, averaged over the machine sizes and the number of instances, are given in Table 7. In the calculation of I_h using Eq. (10), first, the set of efficient solutions of an instance is consolidated among the results of all runs of the respective algorithm and then normalized between [0, 1]. In the calculation of I_ϵ using Eq. (11), the reference set is taken from [4]. Results show the incomparability of the three algorithms in terms of I_h and I_ϵ since the two indicators give contradictory results. However, the simplicity property of MSALS because of its independence of parameter tuning should be taken into account in the comparison of the algorithm with the TP+PLS and RIGP algorithms.

Table 7. Results of average hypervolume and epsilon indicators.

	I_h			I_ϵ		
Job size	MSALS	TP+PLS	RIGP	MSALS	TP+PLS	RIGP
20	1.0682	1.1530	1.1514	1.0066	1.0079	1.0071
50	1.0721	1.2868	1.2612	1.0089	1.0140	1.0116
100	1.0425	1.2786	1.2585	1.0024	1.0063	1.0032
200	0.9618	1.2186	1.2283	1.0002	1.0186	1.0027
Average	1.0434	1.2358	1.2245	1.0050	1.0110	1.0065

6. Conclusion

This study proposes MSALS, a modified version of SALS with two additional acceptance rules, for biobjective PFSS problems to minimize C_{max} and TFT. The main characteristic of SALS is its simplicity in terms of free parameter tuning, and MSALS also possesses this simplicity. Many state-of-the-art algorithms have been proposed for multiobjective PFSS problems, which perform very well according to various multiobjective evaluation criteria. However, experiments conducted on the well-known benchmark instances of [31] intend to compare MSALS with the state-of-the-art algorithms that generate the best existing efficient solutions on the benchmark instances. In comparative study, performance indicators frequently utilized in the multiobjective optimization problems such as the number of efficient solutions found, contributions to the Pareto fronts, ratio of the number of nondominated solutions obtained by an algorithm to the number of efficient solutions of another algorithm, or area covered in the Pareto fronts are considered. The experimental study showed that MSALS is superior to the MOACA, MOSA, and MOIGS algorithms, proposed before in the related literature, in terms of all performance indicators of multiobjectivity. MSALS also yields the best results in terms of the ratio of nondominated solutions to the number of efficient solutions compared to all algorithms included in the comparative study. Coverage of MSALS over other algorithms, except TP+PLS, is greater than that of other algorithms. On the other hand, MSALS exhibits similar performance to BMSA according to the number of efficient solutions and with RIGP according to the average nondominated solutions. Consequently, the MSALS algorithm is either superior to or incomparable with the current best state-of-the-art algorithms in the literature with respect to several multiobjective evaluation criteria. However, it should be noted that MSALS is the simplest algorithm among all available algorithms of the biobjective PFSS to apply any instance without any effort of parameter tuning.

As future research, MSALS can be further improved to provide different learning mechanisms for rule selection. MSALS can also be applied to different multiobjective scheduling problem types to investigate its effectiveness for general multiobjective optimization.

References

[1] Minella G, Ruiz R, Ciavotta M. Restarted iterated Pareto greedy algorithm for multi-objective flowshop scheduling problems. Computers & Operations Research 2011; 38 (11): 1521-1533.

[2] Fowler JW, Kim B, Carlyle WM, Senturk Gel E, Horng SM. Evaluating solution sets of a posterior solution techniques for bi-criteria combinatorial optimization problems. Journal of Scheduling 2005; 8: 75-96.

[3] Yenisey M, Yagmahan B. Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends. Omega 2014; 45: 119-135.

[4] Minella G, Ruiz R, Ciavotta M. A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS Journal on Computing 2008; 20 (3): 451-471.

[5] Coy SP, Golden BL, Runger GC, Wasil EA. Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics 2000; 7: 77-97.

[6] Adenso-Diaz B, Laguna M. Fine-tuning of algorithms using fractional experimental design and local search. Operations Research 2006; 54 (1): 99–114.

[7] Bartz-Beielstein T. Experimental Research in Evolutionary Computation: The New Experimentalism Natural Computing Series. Berlin, Germany: Springer Verlag, 2006.

[8] Dobslaw F. A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In: Proceedings of the International Conference on Computer Mathematics and Natural Computing; Rome, Italy; 2010. pp. 1–4.

[9] Arin A, Rabadi G, Unal R. Comparative studies on design of experiments for tuning parameters in a genetic algorithm for a scheduling problem. International Journal of Experimental Design and Process Optimisation 2011; 2 (2): 103–124.

[10] Birattari M, Stutzle T, Paquete L, Varrentrapp K. A racing algorithm for configuring metaheuristics. In: GECCO 2002 Proceedings of the Genetic and Evolutionary Computation Conference; New York, NY, USA; 2002. pp. 11–18.

[11] Balaprakash P, Birattari M, Stutzle T. Improvement strategies for the F-Race algorithm: sampling design and iterative refinement. In: Bartz Beielstein T (editor). 4th International Workshop on Hybrid Metaheuristics, Proceedings of Lecture Notes in Computer Science. Berlin, Germany: Springer Verlag, 2007. pp. 108–122.

[12] Barbosa EBM, Senne ELF, Silva MB. Improving the performance of metaheuristics: an approach combining response surface methodology and racing algorithms. International Journal of Engineering Mathematics 2015; 2015: 167031. doi: 1155/2015/167031

[13] López-Ibáñez M, Dubois-Lacoste J, Pérez Cáceres L, Stützle T, Birattari M. The irace package: iterated racing for automatic algorithm configuration. Operations Research Perspectives 2016; 3: 43-58.

[14] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science; Nogaya, Japan; 1995. pp. 39–43.

[15] Nannen V, Eiben AE. A method for parameter calibration and relevance estimation in evolutionary algorithms. In: Proceedings of Genetic and Evolutionary Computation Conference; Seattle, WA, USA; 2006. pp. 183–190.

[16] Meissner M, Schmuker M, Schneider G. Optimized particle swarm optimization (OPSP) and its application to artificial neural network training. BMC Bioinformatics 2006; 7: 125. doi: 10.1186/1471-2105-7-125

[17] Hutter F, Hoos H, Stutzle T. Automatic algorithm configuration based on local search. In: Proceedings of the Twenty-Second Conference on Artificial Intelligence; Vancouver, Canada; 2007. pp. 1152–1157.

[18] Hutter F, Hoos HH, Leyton-Brown K, Stutzle T. ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 2009; 36: 267–306.

[19] Smit SK, Eiben AE. Comparing parameter tuning methods for evolutionary algorithms. In: IEEE Congress on Evolutionary Computation; Trondheim, Norway; 2009. pp. 399–406.

[20] Neumüller C, Wagner S, Kronberger G, Affenzeller M. Parameter meta-optimization of metaheuristic optimization algorithms. In: Proceedings of the 13th International Conference on Computer Aided Systems Theory; Las Palmas de Gran Canaria, Spain; 2011. pp. 367-374.

[21] Eiben AE, Smit SK. Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation 2011; 1 (1): 19-31.

[22] Ren Z, Jiang H, Xuan J, Luo Z. Hyper-heuristics with low level parameter adaptation. Evolutionary Computation 2012; 20 (2): 189-227.

[23] Hansen N. An analysis of mutative σ-self-adaptation on linear fitness functions. Evolutionary Computation 2006; 14 (3): 255–275.

[24] Eiben AE, Michalewicz Z, Schoenauer M, Smith JE. Parameter control in evolutionary algorithms. In: Lobo F, Lima CF, Michalewicz Z (editors). Parameter Setting in Evolutionary Algorithms. Berlin, Germany: Springer, 2007. pp. 19-46.

[25] Meignan D, Koukam A, Creput JC. Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism. Journal of Heuristics 2010; 16 (6): 859–879.

[26] Serpell MC, Smith JE. Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms. Evolutionary Computation 2010; 18 (3): 491–514.

[27] Stützle T, López-Ibáñez M, Pellegrini P, Maur M, Montes de Oca MA et al. Parameter adaptation in ant colony optimization. In: Hamadi Y, Monfroy E, Saubion F (editors). Autonomous Search. Berlin, Germany: Springer, 2012. pp. 191-215.

[28] Alabas-Uslu C, Dengiz B. A self-adaptive heuristic algorithm for combinatorial optimization problems. International Journal of Computational Intelligence Systems 2014; 7 (5): 827-852.

[29] Dengiz B, Alabas-Uslu C. A self-tuning heuristic for design of communication networks. Journal of the Operational Research Society 2015; 66 (7): 1101-1114.

[30] Alabas-Uslu C. A self-tuning heuristic for a multi-objective vehicle routing problem. Journal of the Operational Research Society 2008; 59 (7): 988-996.

[31] Taillard E. Benchmarks for basic scheduling problems. European Journal of Operational Research 1993; 64 (2): 278-285.

[32] Varadharajan TK, Rajendran C. A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. European Journal of Operational Research 2005; 167 (3): 772-795.

[33] Arroyo JEC, Armentano VA. Genetic local search for multi-objective flowshop scheduling problems. European Journal of Operational Research 2005; 167 (3): 717-738.

[34] Armentano VA, Claudio JE. An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem. Journal of Heuristics 2004; 10 (5): 463-481.

[35] Framinan JM, Leisten R. A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. OR Spectrum 2008; 30 (4): 787-804.

[36] Rajendran C, Ziegler H. Computational Intelligence in Flow Shop and Job Shop Scheduling: A Multi-Objective Ant-Colony Algorithm for Permutation Flowshop Scheduling to Minimize the Makespan and Total Flowtime of Jobs. Berlin, Germany: Springer, 2009.

[37] Dubois-Lacoste J, López-Ibáñez M, Stützle T. A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Computers & Operations Research 2011; 38 (8): 1219-1236.

[38] Lin SW, Ying KC. Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm. Computers & Operations Research 2013; 40 (6): 1625-1647.

[39] Blot A, Kessaci MÉ, Jourdan L, Hoos HH. Automatic configuration of multi-objective local search algorithms for permutation problems. Evolutionary Computation 2019; 27 (1): 147-171.

[40] Sanjeev Kumar R, Padmanaban KP, Rajkumar M. Minimizing makespan and total flow time in permutation flow shop scheduling problems using modified gravitational emulation local search algorithm. Proceedings of the Institution of Mechanical Engineers Part B: Journal of Engineering Manufacture 2018; 232 (3): 534-545.

[41] Almeida C, Gonçalves R, Venske S, Lüuders R, Delgado M. Multi-armed bandit based hyper-heuristics for the permutation flow shop problem. In: 7th Brazilian Conference on Intelligent Systems; São Paulo, Brazil; 2018. pp. 139-144.

[42] Dengiz B, Alabas-Uslu C, Dengiz O. Optimization of manufacturing systems using a neural network metamodel with a new training approach. Journal of operational Research Society 2009; 60 (9): 1191-1197.

[43] Zitzler E, Thiele L, Laumanns M, Fonseca CM, Da Fonseca VG. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on evolutionary computation 2003; 7 (2): 117-132.