Local search based methods for scheduling in the unrelated parallel machines environment

Lucija Ulaga[a], Marko Đurasević[b,*], Domagoj Jakobović[b]

lucija.ulaga@fer.hr, marko.durasevic@fer.hr, domagoj.jakobovic@fer.hr

[a] *AVL, Zagreb, Croatia*
[b] *University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia*

[*] Corresponding author

Abstract

In many real-world situations, it is necessary to make timely scheduling decisions. In most cases, metaheuristic algorithms are used to solve various scheduling problems because of their flexibility and their ability to produce satisfactory results in a short time. In recent years, several novel or hybrid metaheuristics have been proposed for scheduling problems. Although such research leads to new insights, it inevitably causes certain problems. First, it becomes unclear which methods perform best, especially if they are not properly compared with existing ones. Second, the proposed methods become increasingly complex, making them more difficult to understand and apply. The goal of this study is to investigate the possibility of defining efficient but simple iterative local search (ILS) methods for the parallel unrelated machines environment with minimization of the total weighted tardiness. To improve the efficiency of ILS methods, several design decisions, such as the construction of the initial solution and choice of local search operators. The proposed methods have been compared with several metaheuristics, of which they achieve significantly better results. Thus, we conclude that it is not necessary to increase the complexity of metaheuristics to achieve better results. Rather, better results can be obtained with simple but well-designed local search methods.

Keywords:
Scheduling, Unrelated machines environment, Metaheuristic, Local search, Iterated local search, Total weighted tardiness

1. Introduction

Scheduling processes are important in many real-world situations, such as planning in manufacturing plants (Kofler et al., 2009; Ouelhadj & Petrovic, 2009), universities and schools (Lewis et al., 2007), airports (Cheng et al., 1999; Hansen, 2004), hospitals (Burke et al., 2004; Petrovic & Castro, 2011), and many others. In such problems, the goal is to assign a certain number of jobs or tasks to a limited number of machines or resources in such a way that all required constraints are satisfied and a certain user-defined criterion is optimised (Pinedo, 2012). Solving most scheduling problems is difficult because they are NP hard. Exact methods can provide optimal solutions to such problems Fanjul-Peyro et al. (2019), but they cannot be applied to large-scale problems because it is not possible to enumerate the entire search space. Therefore, most research focused on the development and use of methods that do not necessarily provide optimal results, but can achieve good solutions in a reasonable amount of time. The studies mainly focused on two types of methods, approximation methods (Lenstra et al., 1990) and heuristic methods (Morton & Pentico, 1993). Approximation methods provide some guarantee of optimality of the solution (Pinedo, 2012; Wotzlaw, 2012), while heuristic methods provide no guarantees.

Of the previous two types of methods, heuristic algorithms have attracted considerable attention because they can be adapted to a wider range of problems and are easier to design. Heuristic methods used for solving scheduling problems can be divided into problem-specific heuristics and metaheuristics. Problem-specific heuristics usually appear in the form of dispatching rules (DRs) (Maheswaran et al., 1999; Braun et al., 2001; Đurasević & Jakobović, 2018). DRs are simple heuristics that build the schedule iteratively by deciding which scheduling decision should be made next, i.e., which job should be scheduled on which machine. Because of this, they have limited visibility into the problem, which affects their performance. They can generate the schedule extremely fast, which makes them ideal for dynamic scheduling problems. On the other hand, metaheuristic algorithms search as much of the solution space as possible by starting with complete solutions and iteratively improving them by introducing changes (Hart et al., 2005). Such methods can achieve extremely good results because they do not construct the schedule greedily, but rather try to improve the existing ones.

Various metaheuristics have been applied to the problem of scheduling on parallel unrelated machines. Researchers mainly used genetic algorithms (GAs) Holland (1992), tabu search (TS) (Glover, 1990), simulated annealing (SA) (Kirkpatrick et al., 1983), ant colony optimisation (ACO), variable neighbourhood search (VNS), variable neighbourhood descent (VND) Mladenović & Hansen (1997), greedy randomised adaptive search procedure (GRASP) Feo & Resende (1995), and others. Among such a large number of methods, it is difficult to choose the one that is most appropriate for the problem under consideration. Moreover, the above algorithms are often combined into sophisticated methods that are difficult to understand and reproduce. In many cases, novel metaheuristics are also proposed to deal with scheduling problems. However, such methods usually offer limited novelty (Sörensen, 2015) or the problem in question could have been solved more effectively with a simpler method that yields competitive results (Molnar et al., 2016).

Local search (LS) operators are popular methods for searching in the neighbourhood of the current solution. They have been used in various optimisation problems, including unrelated machines. Due to their increasing popularity, a wide range of LS operators have been proposed in the literature, along with metaheuristics that define how they should be applied. In addition, hybrid algorithms are proposed to further improve performance, which increases their complexity. This inevitably leads to the situation where it is difficult to choose the right LS operators or to determine which strategies should be used to apply them to a problem. However, instead of developing new LS operators or hybrid methods, one should consider if with already existing operators it is possible to construct simple methods that produce high-quality results.

This paper investigates whether various LS operators can be combined into simple but efficient methods for solving the problem of scheduling parallel unrelated machines. For this purpose, different approaches from the literature were analysed, based on which the most important parts of LS methods were identified, such as initial solution generation, perturbation and LS operators.

Based on these observations, two simple iterative LS methods were defined. To provide an exhaustive analysis of the proposed methods, several LS operators were selected from the literature and used as building blocks for the proposed methods. The studied methods were compared with several metaheuristics and the results showed that the iterative LS procedures easily outperform all of the methods. These results indicate that simple LS based methods are as powerful as the more complex metaheuristics. This conclusion is important because it shows that it is more beneficial to focus on constructing simple LS based methods than to apply metaheuristics that may not be appropriate for the problem. The contributions of this work can be summarised as follows:

1. Definition of two simple iterative LS methods and identification of the key design decisions in their development.
2. Analyse the effectiveness of the various LS operators proposed in the literature.
3. Investigation of the influence of different design decisions (initial solution generation, solution acceptance criterion, application of path relinking) on the performance of the simple iterative LS methods.
4. Analysis of the performance of the simple iterative local search methods compared to several previously applied metaheuristics.

The paper is organised as follows. The literature review of heuristic methods applied to the unrelated machines environment problem is given in Section 2. Section 3 contains the description of the unrelated machines scheduling environment. The overview of the methods applied to solve the above problem is given in section 4. Section 5 describes the experimental setup and outlines the obtained results. Finally, Section 6 concludes the paper and provides guidelines for future research.

2. Literature overview

Until now, a large number of studies have addressed the application of metaheuristics to the unrelated machines scheduling problem. Glass et al. (1994) applied GA, SA, and TS to optimise the makespan criterion. SA and TS used two neighbourhood operators, job reassignment and job interchange. The reassignment neighbourhood moved an job from one machine to another, while the interchange neighbourhood swapped two jobs on different machines. The results show that all three algorithms perform similarly. TS was also applied by Srivastava (1998) to optimise the makespan criterion. It uses a single neighbourhood operator that reassigns a task to all other positions in the schedule and selects the best solution. Kim et al. (2002) use SA with six operators to construct the neighbourhood of the current solution. Since the authors consider batches of jobs, they use operators that insert and exchange jobs and batches, and operators that merge and split batches of jobs. The obtained results show that these operators improve the performance of the SA method. The unrelated machines problem with batches is considered by Kim et al. (2003), where SA

is applied with other heuristics to optimise the total weighted tardiness objective. However, the method uses only a single LS operator that exchanges jobs belonging to different batches or machines. Kim & Shin (2003) use a restricted TS, which excludes non-effective job moves from the search. The tested method performs better than some others such as SA or the rolling horizon heuristic. The problem of scheduling printed circuit boards on unrelated machines using a composite GA is studied by Hop & Nagarur (2004).

TS was also used by Logendran et al. (2007) with several initial solution construction methods similar to DRs. The job insert and job swap LS operators were also used. Raja et al. (2008) used a combination of fuzzy logic and a GA, to optimise the total earliness and tardiness criterion. The method used proved to be very effective compared to other methods. A combination of ACO with SA and VNS was proposed by Behnamian et al. (2009). The authors apply 3 neighbourhood operators: job swap on the same machine, job swap between different machines, and insertion of a job on another machine. Based on these neighbourhood operators, they define LS operators that use a single neighbourhood operator but apply it to all jobs in the schedule and select the best result. The hybrid algorithm performed better than any of its individual components. A competitive evolution strategy was used by Chyu & Chang (2010) to optimise the total weighted tardiness and flowtime criteria. A GA, which uses matrix solution coding with new crossover and mutation operators, was proposed by Balin (2011). Chang & Chen (2011) combine the GA with dominance properties that allow the algorithm to obtain near-optimal solutions and outperform other similar methods.

ACO is used by Arnaout et al. (2009) along with a LS operator to improve its performance. In each iteration, the local search operator either swaps two jobs or moves a certain number of jobs to different machines. A variable neighbourhood descent (VND) method is proposed by Fanjul-Peyro & Ruiz (2010) along with several new LS operators that improve its performance. The authors apply the swap and insertion LS operators to search for a better solution, but they also apply three perturbation operators to perturb the solution and avoid local optima. Fanjul-Peyro & Ruiz (2011) also propose a size reduction method to further improve the solutions for the makespan criterion. In addition, Fanjul-Peyro & Ruiz (2012) consider a variant of the problem where it is not necessary to execute each job and where not every machine may be suitable for running every job. A novel method called "GARP", which is a combination of GA with VND and path relinking, was proposed by Haddad et al. (2012). The VND method applies LS operators that swap jobs on the same and different machines, as well as an operator that inserts a job on another machine. However, the operators are applied to all combinations of jobs, and the best obtained solution is selected.

The parallel unrelated machines environment with limited human resources is considered by Costa et al. (2013), where the authors use permutation coding to represent solutions. The TS method is used by Lee et al. (2013) to minimise the total tardiness criterion. The authors use machine list encoding and propose eight LS operators to obtain neighbouring solutions. These LS operators per-

form operations on individual jobs, such as swap or insert, as well as on groups of jobs assigned to the same machine. The ACO algorithm has been applied by Lin et al. (2013) along with 3 different LS operators (job swap on the same machine, job swap between different machines, and job insertion) to improve its performance. A hybrid GRASP method is proposed by de C. M. Nogueira et al. (2014) and achieves good performance compared to other similar methods. The GRASP method uses only the job insertion operator to generate neighbours, but also includes an iterated greedy method to perturb the solutions. The method is also combined with path relinking to improve results. The AIRP method, based on LS, VND and path relinking, is proposed by Cota et al. (2014). The method uses three local search operators: job swap on the same machine, job swap between different machines, and job insertion.

A combination of GA, ACO, and SA is proposed by Afzalirad & Rezaeian (2016). Several metaheuristics, including GAs and bee algorithms, are applied by Rambod & Rezaeian (2014) to address the problem of scheduling parallel unrelated machines with rework processes, machine eligibility, and setup times. Both methods apply a swap local search in their genetic operators. Five hybrid metaheuristics, including ACO and TS, are evaluated by Liao et al. (2014). These methods are applied to solving the unrelated machines scheduling problem with inbound truck sequencing, and therefore the methods and LS operators have been adapted for such a problem. The paper draws conclusions about the situations in which each algorithm variant achieves the best results. A GA with the permutation and floating point representation was applied by Đurasević & Jakobović (2016) and the performance was compared with several simple DRs. Scheduling parallel unrelated machines with heterogeneous delivery vehicles using a single-stage GA is considered by Joo & Kim (2017). Arık (2019) applied the GA, SA and artificial bee colony (ABC) algorithms, of which the ABC achieved the best results. All methods used a simple LS operator which swapped two random jobs. The influence of population initialization in GAs by DRs was investigated by Vlašić et al. (2019), showing that the use of initial solutions generated by DRs can lead to a significant improvement in results. Al-Harkan & Qamhan (2019) propose a two-stage hybrid VNS method combined with SA. Five local search operators were applied, including swapping random jobs on the same machine, swapping two jobs on different machines, swapping all jobs between two machines, swapping the order of jobs, and inserting a job to a different machines. The proposed method achieved better results compared to other methods. An enhanced symbiotic organisms search was proposed by Ezugwu (2019) for the unrelated machines problem with setup times. Finally, seven solution representations for the unrelated machines environment were compared by Vlašić et al. (2020), showing that GA achieved the best results for permutation-based representations.

3. The unrelated parallel machines scheduling problem

3.1. Problem definition

The scheduling problem considered in this paper can be classified as $R|r_j|Twt$ (Pinedo, 2012), which means that the considered environment is the parallel unrelated machines environment with job release times and the total weighted tardiness criterion is optimised. A single problem instance in this environment consists of n jobs, where each job must be scheduled on one of the m available machines. It is assumed that the number of jobs and machines is finite. The subscript j refers to jobs, while the machines are denoted by the subscript i. The basic properties that need to be defined for this problem are job processing times, the release times, the due dates, and the job weights. The processing time of job j on machine i is denoted as p_{ij}. The release time r_j of a job defines the time at which the job is released in the system and can be scheduled. The due date d_j of job j defines the time by which the job needs to be executed. The due date is a soft constraint, i.e., even if some orders are completed after their due date, the schedule is still feasible. However, the delay creates a certain penalty that must be minimised. The weight w_j of a job indicates how important each job and is taken into account in the optimisation criteria. It is also important to mention that the problem was considered under static and deterministic conditions. This means that all the property values of jobs are known in advance and their values do not change.

After a schedule is created based on the above job characteristics, the completion time C_j of each job can be observed. In addition, for each job j, the tardiness can be calculated as $T_j = \max(0, C_j - d_j)$. The tardiness of a job indicates the time that the job was executed after its due date. If the job was completed before its due date, the tardiness of that job is 0. Based on the individual tardiness values, the total weighted tardiness (Twt) of the schedule can be calculated as $Twt = \sum_j w_j T_j$ (Đurasević & Jakobović, 2018). In this criterion, each tardiness value is additionally multiplied by the corresponding job weight to prioritise specific jobs. The goal of this study is to minimise the Twt of the schedule.

To better outline the problem under consideration, the computation of Twt is demonstrated on a small example. Table 1 outlines job properties of an instance consisting of 8 jobs and 3 machines. An example solution for this problem instance is shown in Figure 1 in the form of a Gantt chart that indicates when each job begins and ends its execution. From the schedule, it can be seen that jobs 7, 3, 4, 2, and 6 finished before their respective due dates. Therefore, their tardiness is 0 and they do not contribute to the Twt criterion. This is because the tardiness of jobs can only be positive, which only applies to jobs that are late. For example, job 7 has completed its execution at time 41, while its due date is 70. Therefore, its tardiness would be calculated as follows: $T_7 = \max(41 - 70, 0) = \max(-29, 0) = 0$. Jobs 0, 1, and 5 did not finish by their specified due dates and incur a penalty. For example, job 5 started execution at time 82, and since its processing time on machine 0 is 17, it finished its execution at time 99. Since this job should have been completed by time

Table 1: An example problem instance for the unrelated parallel machines scheduling environment

Job j index	0	1	2	3	4	5	6	7
r_j	62	85	12	44	68	62	43	0
d_j	105	98	60	75	103	91	65	70
w_j	0.97	0.65	0.58	0.56	0.87	0.15	0.95	0.40
p_{0j}	40	95	54	38	42	17	43	41
p_{1j}	40	4	69	60	30	36	89	43
p_{2j}	36	12	48	24	79	91	13	65

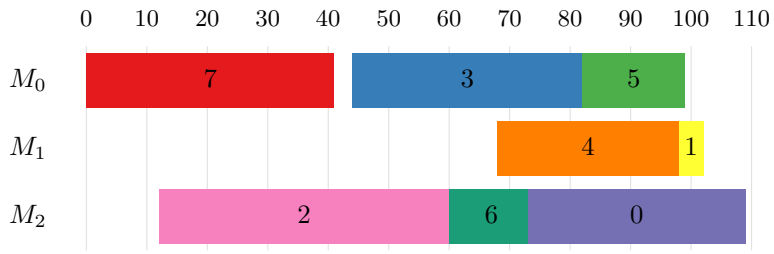


Figure 1: An example solution for the considered problem instance

91, its tardiness is equal to $T_5 = \max(99 - 91, 0) = 8$. A similar calculation can be made for the other two jobs. Since job 1 was completed at time 102, its tardiness is equal to $T_1 = \max(102 - 98, 0) = 4$. Similarly, job 0, whose execution was completed at time 109 but whose due date is 105, has tardiness equal to $T_0 = \max(109 - 105, 0) = 4$. In the end, the total weighted tardiness is calculated by multiplying the tardiness values by the corresponding job weights:

$$Twt = w_0T_0 + w_1T_1 + w_2T_2 + w_3T_3 + w_4T_4 + w_5T_5 + w_6T_6 + w_7T_7$$

$$Twt = 0.97 \cdot 4 + 0.65 \cdot 4 + 0.58 \cdot 0 + 0.56 \cdot 0 + 0.87 \cdot 0 + 0.15 \cdot 8 + 0.95 \cdot 0 + 0.40 \cdot 0$$

$$Twt = 7.68$$

The parallel unrelated machines scheduling problem can formally be defined using a mixed integer programming formulation (Unlu & Mason, 2010; Đurasević & Jakobović, 2020). The time horizon is discretized into time periods $1, \ldots, l$, where l denotes the largest completion time of any job. The binary variable χ_{ij}^t is equal to 1 if job $j \in J$ starts execution on machine $i \in M$ at time

t, otherwise it is equal to 0.

$$\min \quad \sum_{j} w_j T_j \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in M} \sum_{t=0}^{l} \chi_{ij}^{t} = 1 \qquad j \in J \tag{2}$$

$$\sum_{j \in J} \sum_{h=\max(0,t-p_{ij})}^{i-1} \chi_{ij}^{h} \leq 1 \qquad i \in M, t = 1, \ldots, l \tag{3}$$

$$\sum_{i \in M} \sum_{t=0}^{r_j - 1} \chi_{ij}^{t} = 0 \qquad j \in J \tag{4}$$

$$C_j \geq \sum_{i \in M} \sum_{t=0}^{l-1} (t + p_{ij}) \chi_{ij}^{t} \qquad j \in J \tag{5}$$

$$T_j \geq C_j - d_j \qquad j \in J \tag{6}$$

As mentioned earlier, the objective is to minimise the total weighted tardiness of the schedule. Constraint (2) ensures that each job is executed on a single machine at each time. Constraint (3) constrains that a machine executes only a single job at any time. Constraint (4) ensures that no job is executed before it is released in the system. Constraints (5) and (6) represent the constraints imposed on the completion time C_j and tardiness T_j of job j.

3.2. Solution encoding

An important decision when considering the unrelated machines scheduling problem is how to represent the solutions. Various encodings have been proposed in the literature to represent schedules for this problem, including permutation (Costa et al., 2013; Đurasević & Jakobović, 2016), floating point (Bean, 1994; Behnamian et al., 2009), and matrix encodings (Balin, 2011). In this work, the machine list encoding (MLE) (Vallada & Ruiz, 2011) is used. This encoding represents the solution such that each machine has a list of jobs that are associated with it. The jobs in these lists are enumerated in the order in which they are executed on the machines. Figure 2 shows an example solution encoded with MLE for a problem consisting of 3 machines and 10 jobs. In the example, machine 0 is assigned four jobs that must be executed in the specified order. This means that job 5 is executed first, followed by jobs 1, 4, and 9 (in that order). This encoding is used because it is simple, many LS operators have been defined for it, and algorithms obtain good results when they use it (Vlašić et al., 2020).

3.3. Dispatching rules

A plethora of DRs have been proposed for the parallel unrelated machines environment (Đurasević & Jakobović, 2018). Although their performance is not

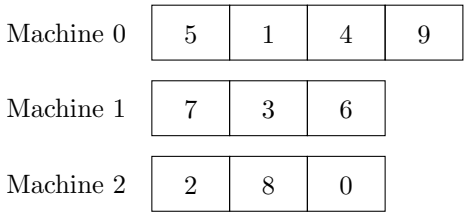


Figure 2: Schedule encoded by MLE

comparable to that of metaheuristics, they can be used to obtain a good initial solution to the problem. The apparent tardiness cost (ATC) rule is one of the most effective DRs used to optimise criteria related tardiness (Lee et al., 1997; Đurasević & Jakobović, 2018). Each time a job is available and a machine is free, all available jobs are ranked according to the following priority function

$$\pi_{ij} = \frac{w_j}{p_{ij}} \exp\left(-\frac{\max\left(d_j - p_{ij} - time, 0\right)}{k\bar{p}}\right),$$

where *time* denotes the current time of the system, $\bar{p}$ denotes the average processing time of all jobs waiting to be scheduled, and k denotes the scaling parameter. Considering the processing time, weight, and time remaining until the job's due date, this DR schedules those jobs for which it receives a higher priority value. The schedule is constructed in a way that each time a scheduling decision must be made, the priority function is calculated for all released jobs and the one with the highest priority value is scheduled. This variant of the rule is suitable for scheduling in dynamic environments, since it does not take into account future information.

In static problems, all information is already available and can be used. Therefore, the ATC rule has been adapted for scheduling under static conditions (Yang-Kuei & Chi-Wei, 2013). Instead of calculating only the priorities of the available jobs, the extended rule calculates the priorities for all jobs. In this way, the rule can determine whether it is better to keep a machine free for a more important job that will arrive in the near future. To account for unreleased jobs, the rule's priority function must also be adjusted:

$$\pi_{i,j} = \frac{w_j}{p_{ij}} \exp\left(-\frac{\max\left(d_j - p_{ij} - \max\left(r_j, time\right), 0\right)}{k_1\bar{p}}\right) \exp\left(-\frac{\max\left(r_j - time, 0\right)}{k_2\bar{p}}\right),$$

where k_1 and k_2 represent the scaling parameters. This version adds an additional term to the priority function that takes into account the time until the job is released in the system, which means that jobs that are released far in the future have a lower priority value.

4. Iterated local search procedures

Over the years, several LS based methods have been proposed, such as GRASP (Feo & Resende, 1995), VNS (Mladenović & Hansen, 1997; Hansen

& Mladenovic, 2001), and VND (Hansen et al., 2016). These methods work in similar ways, with the differences lying in details such as whether or not one or more LS operators are used, or whether or not the LS operators are applied in a deterministic manner. None of these procedures are explicitly used in this work, instead a simple iterated LS (Lourenço et al., 2010) procedure is defined to test the LS operators. The general idea of LS based methods is to start with a given solution to a problem and iteratively improve it by making changes to it. This is usually done by exploring the neighbourhood of the current solution and moving to the best neighbour. The neighbourhood can be explored only once or iteratively until a local optimum is reached. To escape local optima, these methods introduce random changes to the solutions every now and then. In this way, it is possible to jump out of local optima and direct the search to new and possibly better areas.

Based on the previous descriptions, several parts of iterative LS methods can be identified: initial solution generation procedure, perturbation operator, improvement LS operators, and solution acceptance strategy. Based on these elements, the iterative local search (ILS) method is outlined in algorithm 1. The method first generates the initial solution using a selected procedure. Then, several steps are repeated in each iteration until a certain termination criterion is met. First, the current solution is perturbed to introduce random changes. Then, a LS procedure is applied to obtain a new solution. Two ways of improvement are considered. The first, denoted in the algorithm, applies the selected LS operator until convergence. In the second, denoted as CLS, the LS operator is applied only once and the obtained solution can either be accepted or not. It should be noted that methods can use a single LS operator or a set of LS operators. If a set of LS operators is used, one LS operator is randomly selected from the set in each iteration and applied to the current solution. Figure 3 shows the flowcharts of the ILS and CLS variants. The main difference is that CLS applies the LS operator once in each iteration, after which the current solution is updated. In contrast, ILS applies the LS operator until convergence, before testing the acceptance criterion for the obtained solution. All other elements are the same in both variants.

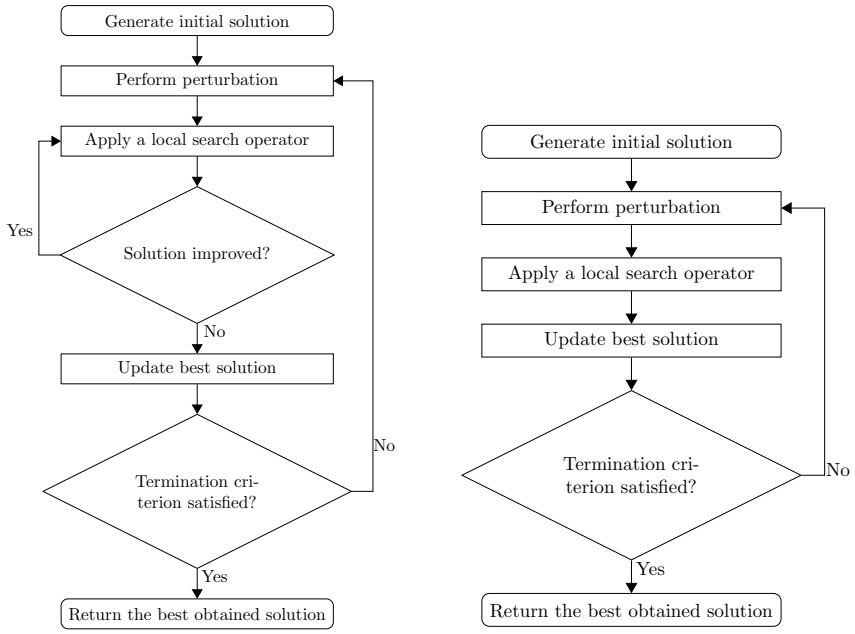
The last item to be specified is the acceptance criterion for the solution obtained after LS. The standard ILS accepts a solution only if it is better than the best solution obtained so far. However, a probabilistic acceptance criterion is also tested. In this acceptance criterion, a better solution is always accepted, but in some cases a worse solution may be accepted. The probability of accepting a worse solution decreases in each iteration by a certain factor (in this case by a factor of 2). This acceptance criterion is inspired by SA, where there is a low probability that a worse solution will be accepted. With this strategy, the algorithm has a greater chance of escaping local optima and starting the search at another solution.

4.1. Initial solution construction

An important part of ILS methods is the construction of initial solutions. In many studies, they are constructed randomly. This can have a negative impact

Algorithm 1 Iterated local search

```
1: S* ← GenerateInitialSolution()
2: while !stoppingCriteria do
3:     S'' ← Perturbation(S*)
4:     do
5:         S' ← S''
6:         S'' ← LocalSearchOperator(S'')
7:     while f(S'') < f(S')
8:     if f(S'') < f(S*) then
9:         S* ← S''
10:    end if
11: end while
12: return S*
```



(a) Flowchart of the ILS variant

(b) Flowchart of the CLS variant

Figure 3: Flowcharts of the two considered iterated local search procedures

on the algorithm, since it has to expend a lot of effort to reach good solutions. On the other hand, starting from good initial solutions can have a significant impact on the results, since the algorithm does not have to invest time to find areas with good solutions, but can immediately start improving them (Vlašić et al., 2019). Therefore, the influence of the initial solution operators was also analysed in this work. For this purpose, five solution initialization techniques were tested:

- Random initialization (random) - the solution is initialised randomly,
- Processing time initialization (processing) - the solution is initialised by placing each job on the machine on which it is executed the fastest (Fanjul-Peyro & Ruiz, 2010),
- Greedy construction initialization (greedy) - the solution is initialised by placing each job in the order of its arrival on the machine for which the optimised criterion would be the lowest. Then, a certain percentage of the jobs with the best criterion values are selected and rescheduled in a random order (de C. M. Nogueira et al., 2014). In this way, the procedure balances between a greedy and a random initialisation,
- ATC initialization - the initial solution is initialised using the ATC DR,
- static ATC initialization - the initial solution is initialised using the static variant of the ATC DR.

4.2. Local search operators

The most important part of ILS is the LS operators used to explore the neighbourhood of a solution. Many operators have been proposed in the literature and have been collected and classified as shown in Figure 4. All operators are divided into two main categories: *perturbation* and *neighbourhood* operators.

The goal of perturbation operators is to change the solution to escape from local optima. These operators do not need to improve the solution, but only introduce random changes in it to seek a different solution space. This is usually done by removing several jobs from the schedule and then reinserting them with a specific strategy. This group of operators consists of:

- Iterated greedy (IG) (Ruiz & Stützle, 2007; Fanjul-Peyro & Ruiz, 2010; de C. M. Nogueira et al., 2014) - a certain number of jobs are removed from the schedule. Each of the removed jobs is reinserted into the schedule at the position that gives the best value for the optimised criterion of the partial schedule. This means that for each job all positions are tested where it can be inserted back into the schedule.
- No same place (NSP) (Fanjul-Peyro & Ruiz, 2010) - selects a random job and places it on another machine that would result in the lowest value of the optimised objective.

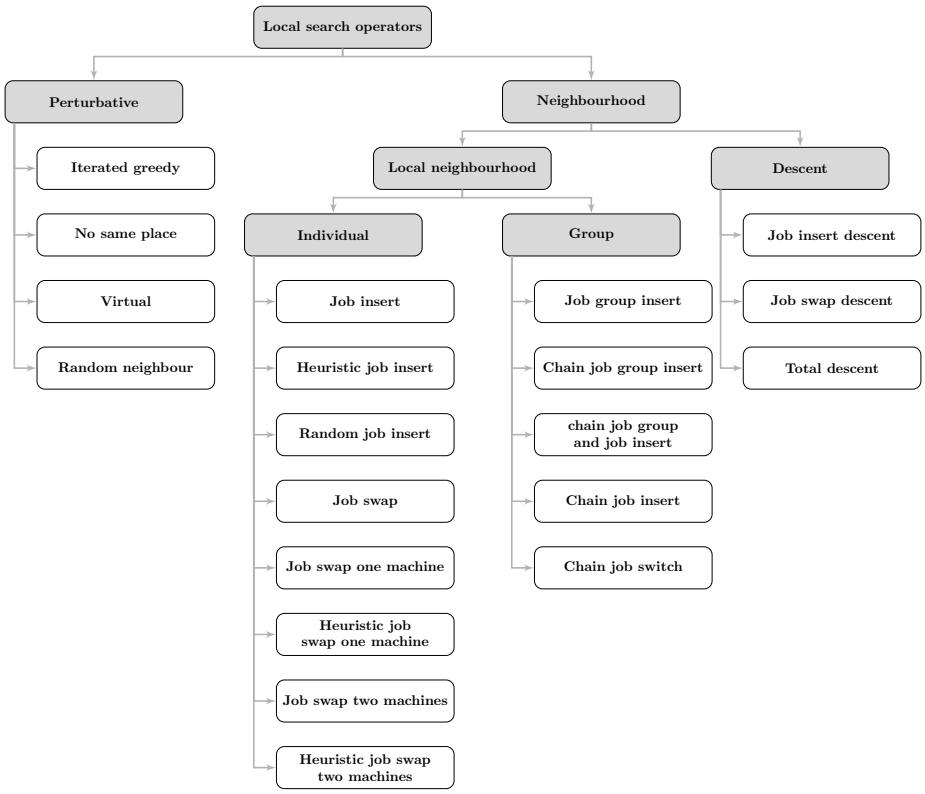


Figure 4: Hierarchical diagram of LS operators considered in this study. Grey nodes represent LS operator groups, while white nodes represent individual LS operators.

- Virtual (VIR) (Fanjul-Peyro & Ruiz, 2010) - works in the same way as NSP, but the job can be placed on the same machine where it was already scheduled.
- Random neighbour (RN) - a random neighbour is created by either inserting job in a different position or swapping two jobs.

The neighbourhood group contains operators that search the neighbourhood of the current solution to obtain new, hopefully better, solutions. The first group *local neighbourhood* contains operators that search only the immediate neighbourhood of the current solution. This means that after searching the neighbourhood, they return the best solution found in that neighbourhood. The operators are further divided into two groups: *individual* and *group* operators. The individual operators search the neighbourhood by modifying a single job at a time, while the group operators perform their operations on a group of jobs. Individual operators include:

- Job insert (JI) (de C. M. Nogueira et al., 2014) - each job is inserted at every possible position and the best solution obtained is selected.

- Heuristic job insert (HJI) (Behnamian et al., 2009) - determines the machines with the highest and lowest Twt values. Each job from the machine with the highest Twt value is inserted at every possible position on the machine with the lowest Twt value. The changes that lead to the best improvement of the total objective value are kept.
- Random job insert (RJI) (Lee et al., 2013) - similar to HJI, but instead of trying every possible combination, a random job is selected from the machine with the highest Twt value and inserted at a random position on the machine with the lowest Twt value.
- Job swap (JS) - each job is swapped with every other job in the schedule. If the swap results in a better solution, the change is kept.
- Job swap one machine (JSOM) (Behnamian et al., 2009) - similar to JS, but the swaps are performed only between jobs on the same machine.
- Job swap two machines (JSTM) (Behnamian et al., 2009) - similar to JS, but the swaps are performed only between jobs on different machines.
- Heuristic JSOM (HJSOM) (Lee et al., 2013) - similar to JSOM, but one swap is performed and the machine and jobs to be swapped are selected randomly.
- Heuristic JSTM (HJSTM) (Lee et al., 2013) - similar to JSTM, but the two machines and jobs on them which will be swapped are randomly selected.

The group operators include:

- Job group insert (JGI) (Lee et al., 2013) - a random group of jobs is selected from the machine with the highest Twt value and inserted on the machine with the lowest Twt value.
- Chain job group insert (CJGI) (Lee et al., 2013) - a random group of jobs will be selected from the machine with the highest Twt value and inserted on a random machine (intermediate machine). A new group of jobs is randomly selected from the intermediate machine and inserted on the machine with the lowest Twt value.
- Chain job group and job insert (CJGJI) (Lee et al., 2013) - similar to CJGI, but only one job is inserted from the intermediate machine to the machine with the lowest Twt value.
- Chain job insert (CJI) (Lee et al., 2013) - similar to CJGI, but jobs are inserted one after another at the intermediate machine and from the intermediate machine to the machine with the lowest Twt value.
- Group job switch (GJS) (Lee et al., 2013) - a random group of jobs is selected from the machine with the highest Twt value and swapped with a random job group from a randomly selected machine that is not the machine with the highest Twt value.

The last group includes descent methods. These operators are executed iteratively until no better neighbour exists. This means that they perform a greedy descent towards an optimum in the given neighbourhood. This group includes:

- Job insert descent (JID) (Fanjul-Peyro & Ruiz, 2010) - inserts each job at every other possible position on another machine in the schedule. When the best solution is found, the procedure is repeated, starting the search at that solution. The search is repeated until no better solution can be found.
- Job swap descent (JSD) (Fanjul-Peyro & Ruiz, 2010) - swaps all pairs of jobs on different machines in the schedule. When the best solution is found, the procedure is repeated, starting the search at that solution. The search is repeated until no better solution can be found.
- Total descent (TD) - a combination of JID and JSD, as it searches both neighbourhoods.

4.3. Path relinking

In several cases, LS operators have been coupled with intensification procedures to obtain better solutions. One such method is path relinking (PR) (Glover et al., 2000), which has often been coupled with other methods in solving scheduling problems. The idea of PR is to construct the path from one solution to another and hopefully obtain a better solution that lies on that path. The best solutions are placed in the elite set, which is a list of good solutions. PR is applied after a solution is obtained by the LS operator by selecting a random solution from the elite set and constructing the path between these two solutions. The path is constructed using job swaps and insertions until the two solutions are equal. The best solution on this path is then returned as the new solution and inserted into the elite set. Since PR has often been used in conjunction with LS based methods, this paper investigates whether basic ILS can be improved by combining it with PR.

5. Results and discussion

5.1. Experimental setup

To test the considered algorithms, a set of problem instances was generated using procedures from the literature (Kim et al., 2003; Lin et al., 2013; Lee et al., 2013; Vlašić et al., 2020). The generated set consists of 60 problem instances with different properties. The instances were generated with different combinations of the number of jobs and machines, with the number of jobs set to 12, 25, 50, and 100 and the number of machines set to 3, 6, and 10. Job processing times were generated from the interval $p_{ij} \in [0, 100]$. The processing times of each job are generated with either a uniform, normal or quasi-bimodal distribution. In this way, it is simulated that jobs are released from different

sources. The weights of jobs were generated from the interval $w_j \in (0, 1]$, where a larger weight means that the job is more important. The job release times are generated uniformly form the interval $r_j \in \left[0, \frac{\hat{p}}{2}\right]$, where $\hat{p}$ is defined as.

$$\hat{p} = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} p_{ij}}{m^2}.$$

Job due dates are generated uniformly from the interval

$$d_j \in \left[r_j + (\hat{p} - r_j) * \left(1 - T - \frac{R}{2}\right), r_j + (\hat{p} - r_j) * \left(1 - T + \frac{R}{2}\right)\right],$$

where T and R are the due date tightness and due date range parameters, respectively. The due date range controls the dispersion of due dates in time. The due date tightness specifies how close to the release times of jobs the due dates are generated, which directly affects the difficulty of the problem. Problem instances were created using both parameters with values of 0.2, 0.4, 0.6, 0.8, and 1 in various combinations.

Each tested method was executed 30 times on all problem instances to obtain statistically significant results. Based on the 30 obtained results, the minimum, median, and maximum values for each method were calculated. The results reported in the experiments represent the Twt value obtained for all 60 problem instances in the set. To determine if there is a statistical difference between the methods tested, the Kruskal-Wallis test was used when comparing a group of methods with the Canover method adjusted by Benjamini-Hochberg for the post-hocc analysis, and the Mann-Whitney test was used for pairwise comparison. The difference in results is considered significant if the obtained p-value is less than 0.05. The termination criterion was set at 5 seconds per instance, as the experiments showed that this amount of time is sufficient for the tested methods to converge. This ensures that each method will perform the same amount of work and that the comparisons are fair. The parameters for all methods from the literature were fine-tuned through preliminary experiments to improve their performance, and the best parameter values were used for the comparisons.

The lower bound (LB) for the problem instances is not known because they are too large to be solved by exhaustive search methods. However, the best result obtained for each instance by any method tested in the experiments was used to calculate the approximation of the lower bound for the considered problem set. This approximated LB was 9.419 and will be used in further sections to analyse how far away the methods are from the LB.

5.2. ILS method analysis

This section analyses the individual components of the ILS method to determine how they affect its performance.

Table 2: Performance of the initial solution procedures

	Random	Greedy	Processing	ATC	Static ATC
Twt	1059	337.5	247.8	13.38	12.38

5.2.1. Analysis of initial solution generation methods

Since LS operators use a single solution that they iteratively improve, the selection of an appropriate initial solution can have a significant impact on the results. Therefore, we first analyse the impact of the quality of the initial solution on the results obtained. The five solution initialization methods produce solutions of significantly different quality, as can be seen in Table 2. Since the greedy and random methods are stochastic, the values in the table represent the median of 30 executions of these methods. Compared to the other methods, the ATC rules produced solutions that were one or two orders of magnitude better.

5.2.2. Analysis of the local search operators

It can be concluded that ATC and static ATC provide the best results, but it is questionable whether this has a significant impact on the LS procedures. Therefore, the immediate neighbourhood of the initial solutions is investigated, which means that each LS operator is applied only once to the generated initial solutions. The results are shown in Table 3, where the cells represent the median values of 30 individual executions. The values in bold denote the best result obtained with each LS operator. Note that the descent operators perform multiple iterations of job insertions or swaps and are therefore expected to produce better results than others. The best neighbours are obtained with the solution generated by the static ATC heuristic. As expected, the results show that several operators could not improve the quality of the schedule for some initial solutions due to randomness in job selection and swapping. The best results are obtained by the descent operators. Among the operators that are applied only once, the JS and JI operators obtained the best solutions. This shows that it is possible to improve the results of most LS operators by searching in the neighbourhood of good solutions.

To further examine the LS operators, each is applied repeatedly to the solutions generated by the initialization procedures with a limit of 5 seconds. Table 4 shows that the operators obtained improved solutions when applied multiple times, giving them the opportunity to achieve better solutions. This can be seen most clearly in the random, greedy and processing initialization methods, where the results were significantly improved compared to the variant where the LS operator was used only once. Still, most of them get stuck in local optima that are relatively far from the LB. This highlights the limitations of using only a single LS operator, since at some point they reach a solution from which they cannot get out. An interesting phenomenon is that the RN operator obtained among the best results, showing that it is possible to get good results even with a random sampling of the search space.

Table 3: Results obtained by applying the LS operators once on starting solutions generated by different initial solution methods

	Random	Greedy	Processing	ATC	Static ATC
Initial	1059	337.5	247.8	13.38	12.38
JI	896.5	284.2	195.2	11.88	**11.22**
TD	13.82	12.22	14.05	10.45	**10.30**
CJGI	1010	335.4	245.9	13.38	**12.38**
CJGJI	999	336.8	244.8	13.38	**12.38**
JGI	1005	336.0	243.8	13.38	**12.38**
GJS	1061	337.7	243.9	13.38	**12.37**
JGS	16.67	16.16	18.41	10.76	**10.47**
JID	31.53	23.35	27.29	**11.15**	11.35
JSD	1010	332.6	242.45	13.38	**12.38**
CJI	1003	331.3	240.8	13.38	**12.36**
HJSOM	1040	334.7	246.4	13.38	**12.38**
HJSTM	1010	334.5	240.9	13.38	**12.37**
RJI	939.4	295.5	212.0	12.82	**12.12**
JSOM	904.1	283.9	186.9	12.58	**11.71**
JSTM	861.8	273.9	194.6	12.18	**11.94**
JS	856.7	274.7	183.9	11.87	**11.48**
NSP	831.6	262.5	198.3	13.11	**12.14**
IG	812.5	252.5	191.7	12.99	**12.10**
RN	1040	332.0	245.2	13.38	**12.38**
VIR	825.7	257.5	195.0	12.88	**12.01**

The results outline the differences in the performance of the applied LS operators. Some LS operators perform well across all the initial solution generation methods, like JI or TD, whereas some still achieve poor results when used with non ATC based initialisation procedures. This demonstrates that there is a significant difference in the performance of LS operators, where some operator types can more easily navigate to good solutions. However, these results do not give a definite overview of the performance of LS operators, since by starting from a single solution it is just a matter of time when they get stuck in a local optima.

Based on the outlined results, it is possible to conclude that the ATC initialisation methods represent the best choice for generating the starting solution. Although the relative improvements are the smallest for this initialisation procedure, all of the LS operators easily achieve the best performance by starting

Table 4: Results obtained by applying the LS operators iteratively (with a limit of 5 seconds per instance) on starting solutions generated by different initial solution methods

	Random	Greedy	Processing	ATC	Static ATC
Initial	1059	337.5	247.8	13.38	12.38
JI	11.80	11.40	10.66	10.30	**10.11**
TD	13.66	12.44	14.05	10.45	**10.30**
CJGI	42.98	31.46	28.42	12.12	**11.19**
CJGJI	37.16	27.45	26.19	12.01	**11.14**
JGI	351.5	116.1	83.17	13.01	**12.15**
GJS	288.8	34.73	14.51	12.14	**11.18**
JGS	17.61	16.51	18.41	10.76	**10.47**
JID	30.82	23.05	27.29	**11.15**	11.35
JSD	51.48	30.06	27.50	11.94	**11.16**
CJI	336.4	115.9	80.09	12.95	**12.15**
HJSOM	41.90	27.86	30.44	12.12	**11.68**
HJSTM	338.0	113.4	80.55	12.96	**12.14**
RJI	236.7	72.75	62.83	12.71	**11.90**
JSOM	282.7	35.08	16.30	12.12	**11.21**
JSTM	37.77	25.79	28.69	**11.36**	11.40
JS	23.74	14.84	13.19	**10.34**	10.44
NSP	20.18	15.40	16.35	10.84	**10.59**
IG	19.78	11.31	11.53	10.05	**9.85**
RN	10.94	10.48	10.54	**9.960**	10.03
VIR	17.21	12.01	11.26	10.53	**10.21**

from them. Therefore, in further sections the static ATC method will be used to generate the initial solution for all experiments.

5.2.3. Analysis of perturbation operators

The previous results show that the individual LS methods cannot achieve results close to the lower bound. This is because they have limited exploratory capability and usually focus only on exploitation, i.e., obtaining the best solution in the neighbourhood. Therefore, to obtain a method that has both exploratory and exploitative capabilities, it is necessary to combine both types of LS operators in the procedure. In this way, the operators of one group can introduce perturbations in the solution to escape local optima, while the others try to find the best solution in the neighbourhood. Therefore, the LS operators that fall into the perturbation group are used to introduce random changes

into the current solutions, while the other LS operators are used to search the neighbourhood of the solution.

To get an idea of the effectiveness of the perturbation operators, each operator is combined with every other LS operator using both the CLS and ILS combination strategies. The median values of these results are shown in Table 5. The values in bold denote the best result obtained for each LS operator. The first thing to note is that the NSP and VIR operators do not lead to good results and perform worse than IG and RN for all LS operators. The reason for this is that NSP and VIR make more conservative changes compared to IG. On the other hand, RN gives the best results when the CLS strategy is used, since it is used more often in this situation and is more conservative (it introduces smaller changes to the solution) compared to IG. Although the RN operator performs better in more situations, it is difficult to assess whether it really performs better than IG, since the performance strongly depends on both the strategy used (CLS or ILS) and the specific LS operator. Therefore, both will be considered in further experiments.

Table 5: Results obtained for different perturbation operators

	CLS				ILS			
	NSP	RN	IG	VIR	NSP	RN	IG	VIR
JI	10.15	9.855	9.684	10.12	10.14	9.635	**9.527**	10.13
TD	10.32	**9.516**	9.706	10.05	10.30	9.626	9.717	10.02
CJGI	10.34	9.704	**9.678**	9.994	11.35	11.47	11.58	11.20
CJGJI	10.35	9.692	**9.674**	10.02	11.14	11.30	11.31	11.00
JGI	10.58	**9.744**	9.855	10.23	11.03	11.36	11.08	10.80
GJS	10.19	9.935	9.827	10.16	9.915	**9.679**	9.857	9.707
HJI	10.46	**9.571**	9.770	10.17	10.47	9.572	9.693	10.18
JID	10.40	9.564	9.713	10.05	10.42	**9.599**	9.685	10.07
JSD	10.35	9.640	**9.585**	9.991	10.71	11.11	10.98	10.57
CJI	10.58	**9.782**	9.844	10.21	10.49	10.98	10.43	10.35
HJSOM	10.56	**9.783**	9.836	10.22	10.53	10.97	10.48	10.36
HJSTM	10.42	**9.614**	9.828	10.09	10.41	10.74	10.58	10.25
RJI	10.56	9.967	**9.845**	10.18	10.55	10.26	9.867	10.18
JSOM	10.08	9.750	9.795	10.02	10.10	**9.614**	9.637	10.03
JSTM	10.21	9.975	**9.807**	10.16	10.32	10.31	9.820	10.19
JS	10.40	**9.717**	9.831	10.05	10.48	10.24	9.910	10.14

Now that the LS operators are used with various perturbation operators, one

can get a better idea of their performance. From these results, it appears that some operators perform much better than others. Thus, the choice of the used operator is very important, as it is clear that certain operators simply do not perform well. This is especially true for operators which manipulate groups of jobs. Some other operators such as the RJI and JSTM operators also performed poorly. On the other hand, the best overall results were obtained by the JI and TD operators. The reason for the good results of these operators is that they perform a simple but thorough search of the neighbourhood. This allows them to easily navigate to good solutions.

5.2.4. Analysis of different LS operator combinations

Although good results can be obtained by using a single LS operator, this could limit the ability of the method to effectively explore the search space and obtain good solutions. Therefore, in this section we analyse how the ILS and CLS methods perform when different combinations of LS operators are used. Although many combinations were tested, only the 10 combinations with the best results, listed in Table 6, are included in the analysis. The perturbation method used is given in the second column and the LS operators in the third column. In each iteration of the algorithms, a LS operator is randomly selected from the given list of operators in the second column and used to generate the neighbourhood. Both strategies, CLS and ILS, are tested with all of the enumerated combinations.

Table 6: Tested ILS combinations

Method ID	Perturbation	LS operators
1	IG	all LS operators except perturbation
2	IG	JID, JSD, HJSTM, CJI
3	IG	all LS operators except IG
4	IG	JI, TD
5	IG	JI, JS
6	RN	JI, TD
7	RN	JI, JS
8	RN	all LS operators except perturbation
9	IG	HJI, JSOM, JSTM
10	IG	TD, HJI, JSOM, JSTM, CJI, HJSTM

The results of the considered LS operator combinations are given in Table 7. The bold values show the best values obtained for each strategy. It is immediately apparent that a wide range of results is obtained for the various LS operator combinations. Therefore, one still has to be very careful when selecting the LS operators. For the CLS method, the best results were obtained for combinations 1, 3, and 8. In these variants, almost all LS operators are used.

This shows that the CLS method prefers a larger set of operators, probably due to the fact that they are applied only once after each perturbation. Therefore, using a larger number of LS operators will further increase the diversity of the obtained solutions. On the other hand, the ILS method obtained the best results for the first six LS operator combinations, especially for combinations 2, 4, and 6. Experiments 4 and 6 are particularly interesting because only two LS operators are used, which seems to be sufficient to obtain good solutions. Since in this strategy the operators are applied until convergence, it seems to be more advantageous to use only a small number of operators. The results also show that the ILS method obtains the best solutions for more combinations compared to the CLS method, which makes it somewhat less sensitive to the choice of LS operators used.

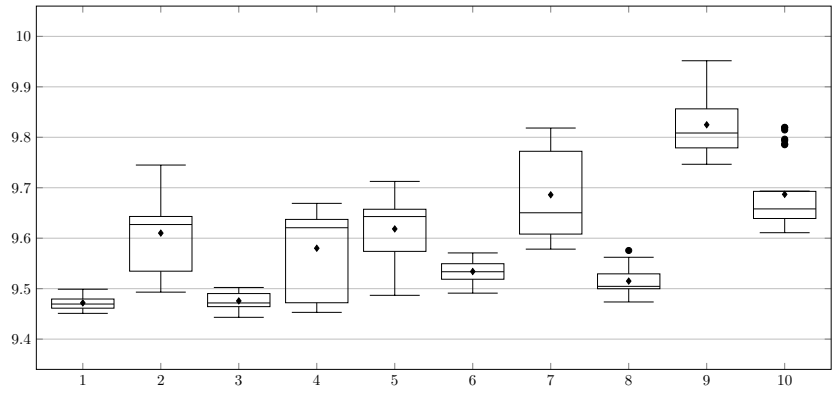
Table 7: Result obtained by different LS operator combinations

	CLS			ILS		
	Min	Med	Max	Min	Med	Max
1	9.451	**9.470**	**9.499**	9.465	9.488	9.602
2	9.493	9.628	9.745	**9.440**	9.473	9.578
3	**9.443**	9.472	9.502	9.467	9.569	9.790
4	9.453	9.623	9.669	9.450	**9.471**	**9.522**
5	9.487	9.644	9.713	9.460	9.493	9.536
6	9.491	9.534	9.571	9.456	9.482	9.528
7	9.578	9.653	9.818	9.603	9.661	9.780
8	9.474	9.505	9.576	9.624	9.669	9.917
9	9.747	9.810	9.952	9.575	9.635	9.832
10	9.611	9.659	9.820	9.627	9.659	9.849

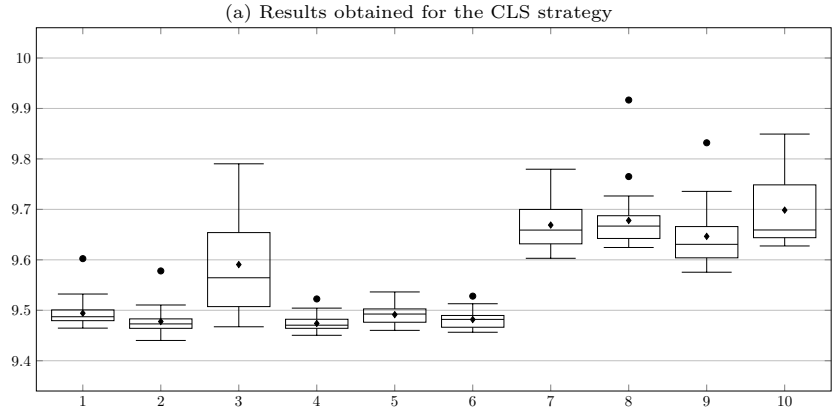
The results are also shown in Figure 5 in the form of boxplots. The two plots illustrate that the two strategies often perform quite differently when using the same LS operator combinations. Moreover, the graphs show that in the cases where poor results were obtained, they were also more scattered and contained multiple outliers. The plots show that the ILS method is more resilient to the choice of LS operators, as the CLS method gives more scattered results in most experiments. However, there is no statistically significant difference between the best results of CLS and ILS, showing that both perform equally well for each best choice of operators.

5.2.5. Analysis of the intensification procedure

To investigate the effectiveness of the PR procedure, it was applied to all LS combinations given in Table 6. The results obtained are shown in Table 8. The bold cells represent the values where the addition of PR led to a better



(a) Results obtained for the CLS strategy



(b) Results obtained for the ILS strategy

Figure 5: Boxplot representation of the results obtained for CLS and ILS

result compared to the method without PR in Table 7. In most cases, the introduction of PR did not lead to better results. On the contrary, in the cases where CLS and ILS gave the best results, such as CLS-1, CLS-3, or ILS-2, the addition of PR led to a significant deterioration of the results. Although in some cases the addition of PR led to better results, statistically these results were rarely better. Only in the ILS-7 method were the results with PR statistically better than those without PR. However, in all other cases, the results obtained with PR were equally good or significantly worse than the results obtained without PR. In general, PR only leads to an improvement in a very limited number of scenarios, and in most cases it did not improve the performance of the algorithm under consideration. The reason for this is probably that the selected LS operator combinations can already cover the search space well, and thus a rather costly intensification method that tries to find better solutions in its region is redundant, as it consumes time that could have been used by the LS operators.

Table 8: Result obtained by different LS operator combinations with PR

	CLS			ILS		
	Min	Med	Max	Min	Med	Max
1	9.500	9.684	9.898	9.472	9.520	9.673
2	9.561	9.771	9.876	9.526	9.585	9.750
3	9.633	9.827	10.10	9.496	9.646	10.20
4	9.501	9.653	9.849	9.472	9.497	9.708
5	9.599	9.734	9.956	9.569	9.624	9.739
6	9.506	9.578	9.702	9.460	9.549	9.811
7	**9.512**	9.682	9.903	**9.470**	**9.507**	**9.561**
8	9.535	9.796	9.953	9.651	9.783	9.942
9	**9.645**	9.868	9.985	9.698	9.785	10.02
10	**9.606**	9.662	9.836	**9.609**	**9.653**	**9.837**

5.2.6. Analysis of the stochastic acceptance criterion

In this section we study the influence of using the stochastic acceptance criterion, which allows a worse solution than the current one to be accepted according to the LS operator. Table 9 shows the ILS and CLS methods with the stochastic acceptance criterion. The bold numbers represent the values where the methods with the stochastic acceptance criterion performed better than the corresponding variants from Table 7. The results show that the stochastic acceptance criterion is able to significantly improve the results in several cases. The stochastic acceptance criterion worked particularly well for the CLS variant, which is most likely due to the fact that it can accept worse solutions more often than ILS. Unlike the case where the standard acceptance criterion was used, here the CLS method achieves better results than the ILS method in more cases. This also shows how a small design decision can have a large impact on the results obtained.

5.3. Comparison with metaheuristic methods

In the last section, various ILS elements were analysed, resulting in several good LS operator combinations. However, it is necessary to test how these methods compare to other approaches that have been widely used in the literature. Therefore, in this section, the best ILS methods from the last section are compared to several metaheuristic methods:

- GA with random population initialization Vlašić et al. (2020) and a population initialised by different dispatching rules (GA +DRs) Vlašić et al. (2019)
- ACO with one LS Arnaout et al. (2009)

Table 9: Result obtained by LS operator combinations with the stochastic acceptance criterion

	CLS			ILS		
	Min	Med	Max	Min	Med	Max
1	**9.441**	**9.459**	**9.494**	**9.449**	**9.473**	**9.534**
2	9.970	10.27	10.54	9.495	9.568	9.938
3	**9.441**	**9.466**	**9.497**	**9.438**	**9.472**	**9.547**
4	**9.450**	**9.468**	**9.653**	9.459	9.483	9.538
5	**9.460**	**9.502**	**9.677**	**9.454**	**9.478**	**9.524**
6	**9.467**	**9.497**	**9.542**	9.611	9.650	9.739
7	**9.523**	**9.591**	**9.632**	9.623	9.664	9.813
8	**9.454**	**9.483**	**9.527**	**9.589**	**9.637**	9.927
9	10.02	10.34	10.62	9.747	9.849	9.928
10	**9.536**	**9.611**	**9.767**	**9.562**	**9.612**	**9.681**

- TS with multiple LS operators (JGI, CJGI, CJGJI, CJI, GJS, HSJOM, HJSTM) Lee et al. (2013)
- GRASP with ILS and PR (GRASP +ILS+ PR) using the IG and JS operatorsde C. M. Nogueira et al. (2014)
- VNS with HJI, JSOM, and JSTM Behnamian et al. (2009)
- VND with VIR, IG, JID, and JSD Fanjul-Peyro & Ruiz (2010)
- SA with JS Behnamian et al. (2009)

These methods were chosen because they represent popular metaheuristics that have been most commonly used to solve the problem under consideration. Moreover, with the exception of ACO, the outlined metaheuristics used the same encoding (MLE) to represent solutions. All methods were adapted to the studied problem and their parameters were fine-tuned.

The seven LS methods that produced the best results are labelled CLS or ILS, depending on the strategy, and the ID from Table 7, which represents the LS combination that was used. The variants using the stochastic acceptance criterion are labelled with the prefix "S" along with the strategy and the ID of the LS combination.

Table 11 presents the results obtained with the considered methods. Among the tested methods, ACO has obtained the worst results, statistically worse than all other methods. Although this method is coupled with a LS operator, the performance was still quite poor. VND, TS, and VNS achieved similar results. Of these three methods, VNS achieved the most widely scattered results and the best median and minimum values, which seems to indicate that it is superior to the other two methods. However, these methods do not achieve the same results

as GA with a randomly initialised population, which performs significantly better. SA achieves significantly better results than either of the previous methods, even though it uses only a simple LS operator. Of the methods based on LS from the literature, GRASP achieved the best overall results. Of the tested methods, GA initialised with DRs achieved the best result. So, it is obvious that the methods from the literature give quite different results, which clearly shows that the choice of method is important for solving this problem.

In order to test whether the observed differences are statistically significant, the Kruskal-Wallis test was performed, which yielded a p-value of 0. This means that there are significant differences between the considered methods. Post-hoc analysis was performed to determine how the methods performed in pairwise comparisons. Table 10 shows the results of the post hoc analysis, where $=$ means that there is no significant difference, $<$ means that the method in the row performs significantly worse than the method indicated in the column, and $>$ means that the method in the row performs significantly better than the method indicated in the column. Based on the obtained results, it is clear that the selected ILS and CLS variants perform significantly better than any of the other tested metaheuristics. On the other hand, except in one case, there is no significant difference between the different ILS and CLS methods. This is further evidence that methods based on LS operators can significantly outperform many existing metaheuristics, and most of them by a large margin. Among the standard metaheuristics, GA+DRs performed significantly better than all other existing metaheuristic methods, followed by GRASP and SA (which performed equally well), then TS, VNS and VND (which again performed statistically equally), and finally ACO performed significantly worse than all other methods.

The proposed ILS achieved significantly better results than all the other methods mentioned above. Among them, ILS-5 obtained the worst results, while the best results were obtained by S-CLS-1 and S-CLS-3, which use the stochastic acceptance criterion. Comparison with the LB shows that the best solution (ILS-2) is only about 0.5% worse. The best solutions and median values of the other LS methods are also quite close to the obtained LB. These results show that the proposed ILS methods can easily outperform the existing algorithms.

Figure 6 shows the results in the form of a boxplot (without ACO, as this would affect readability). Most of the solutions obtained by GA when using DRs to initialise the initial population are better than the best solution obtained by GA when randomly generating the initial population. Except for one outlier, the worst solutions obtained using the LS methods are better than the best result obtained using any other metaheuristic method. Another strength of the methods based on LS is that the dispersion of their solutions is immensely small. For example, the worst solution for S-CLS-1 is no greater than 9.5, and considering that the best objective value achieved by all methods combined is 9.419, the relative difference is less than 1%.

It is also interesting to observe the convergence rate of the algorithms. All methods except ACO are shown in Figure 7 to illustrate how the fitness of the

Table 10: Statistical results of the post hoc analysis for pairwise comparison between the tasted methods

	GA	GA+DRs	ACO	TS	GRASP+ILS+PR	VNS	VND	SA	CLS-1	CLS-3	ILS-2	ILS-4	ILS-6	S-CLS-1	S-CLS-3
GA	-	<	>	=	<	=	>	<	<	<	<	<	<	<	<
GA+DRs	>	-	>	>	>	>	>	>	<	<	<	<	<	<	<
ACO	<	<	-	=	<	<	<	<	<	<	<	<	<	<	<
TS	=	<	>	-	<	=	=	<	<	<	<	<	<	<	<
GRASP+ILS+PR	>	<	>	>	-	>	>	=	<	<	<	<	<	<	<
VNS	=	<	>	=	<	-	=	<	<	<	<	<	<	<	<
VND	<	<	>	=	<	=	-	<	<	<	<	<	<	<	<
SA	>	<	>	>	=	>	>	-	<	<	<	<	<	<	<
CLS-1	>	>	>	>	>	>	>	>	-	=	=	=	=	=	=
CLS-3	>	>	>	>	>	>	>	>	=	-	=	=	=	=	=
ILS-2	>	>	>	>	>	>	>	>	=	=	-	=	=	=	=
ILS-4	>	>	>	>	>	>	>	>	=	=	=	-	=	=	=
ILS-6	>	>	>	>	>	>	>	>	=	=	=	=	-	<	=
S-CLS-1	>	>	>	>	>	>	>	>	=	=	=	=	>	-	=
S-CLS-3	>	>	>	>	>	>	>	>	=	=	=	=	=	=	-

Table 11: Result obtained by different metaheuristic methods

Index	Method	Min	Med	max
1	GA	9.682	9.934	10.15
2	GA+DRs	9.536	9.595	9.670
3	ACO	11.39	11.96	12.43
4	TS	10.01	10.10	10.19
5	GRASP+ILS+PR	9.578	9.724	9.891
6	VNS	9.791	10.05	10.58
7	VND	10.05	10.15	10.34
8	SA	9.576	9.710	9.937
9	CLS-1	9.451	9.470	9.499
10	CLS-3	9.443	9.472	9.502
11	ILS-2	**9.440**	9.473	9.578
12	ILS-4	9.450	9.471	9.522
13	ILS-6	9.456	9.482	9.528
14	S-CLS-1	9.441	**9.459**	**9.494**
15	S-CLS-3	9.441	9.466	9.497

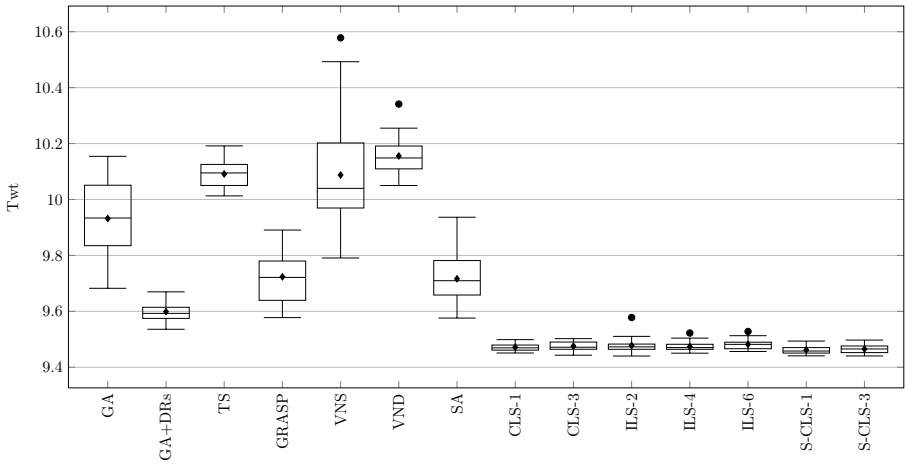


Figure 6: Boxplot representation of the results for different metaheuristics

best solution changes over time. It is obvious that all algorithms have a very similar convergence curve. At the beginning, they all rapidly improve the fitness of the best solution up to about 0.3-0.5 seconds, then the improvement slows down. However, the main difference is in the solution quality, where the methods

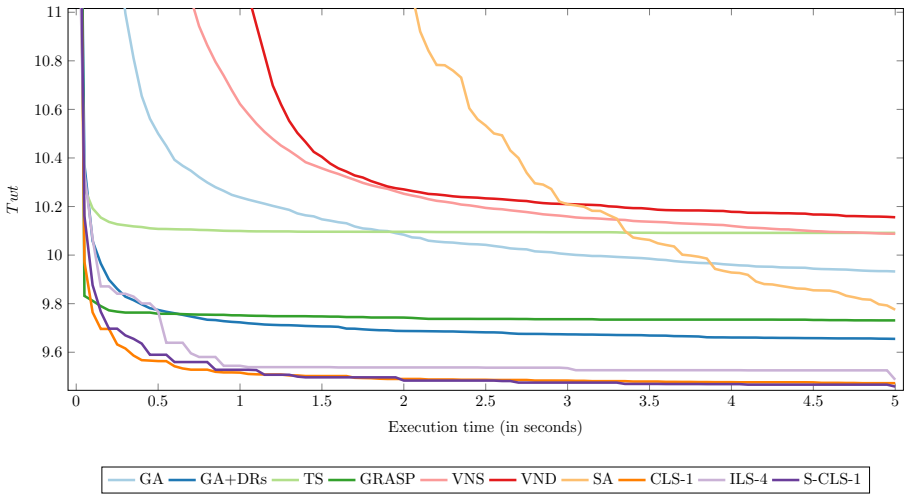


Figure 7: Convergence of different metaheuristic methods

start to stagnate. TS is the worst in this respect, because after obtaining a good solution, it is not able to escape the local optimum and move to better solutions. All GA variants have a very similar convergence curve, but depending on which initialization strategy was used, the curve is lower. All ILS and CLS variants converge fairly quickly and usually overlap in later iterations. In the end, it is obvious that good results can be achieved in a fraction of the time that was available. This could prove important in situations where the time available to create the schedules is limited.

5.4. Discussion

The results in the previous sections show that the proposed simple ILS methods can outperform several existing metaheuristics. However, the most important question is why similar approaches from the literature do not perform as well or perform so poorly.

First of all, an important part in applying metaheuristics to this problem is related to generating a good initial solution. As has been shown, the choice of the starting solution can significantly affect the performance of the LS operators. Although several initial solution methods have been defined in the literature, their results lag behind those of the DRs. This leads to significantly slower convergence, since the algorithms have to invest a lot of time to reach the search space that contains good solutions. Research conducted by Vlašić et al. (2019) has already shown that using DRs as initialization methods leads to significantly better results. Therefore, DRs can be used as a fast and easy method to obtain good initial solutions and thus increase the convergence speed of algorithms.

As for the choice of metaheuristics, it is obvious that more complex algorithms do not necessarily provide better results. A good example of this is ACO, which produced the worst results for several reasons. First, the solution

representation is not "natural" and this limits the use of generating initial solutions and LS operators. Also, there are a large number of parameters that need to be optimised, and poor choice of these parameters can reduce efficiency. Although TS performed better than ACO, it did not perform better than the other algorithms that do not use the concept of tabu lists. The reason is that the housekeeping that must be done when dealing with the tabu list (updating the list, finding tabu moves) takes a lot of time and reduces the number of schedules that can be tested. Therefore, it seems more beneficial to invest this time in finding more neighbours rather than using it to narrow down the neighbourhood. In addition, a large number of LS operators were used, which did not perform well on their own. On the other hand, the best results were obtained with GRASP, SA, and GA. The results for GA and SA are particularly important because both algorithms use simple operations to modify the solution, showing that the methods used do not need to be very sophisticated to obtain good results.

The last point to note is the LS operators used in each method to obtain a new solution. Since VNS, VND, GRASP, SA, and TS all use LS operators, their differences are mostly the result of using different LS operators. For example, VND uses the JSD and JID LS operators in conjunction with the NSP, VIR, and IG perturbation operators. However, initial analysis has shown that these operators generally work best with the RN perturbation operator. Combined with the fact that a poor initial solution was used, this resulted in an overall poor performance of the method. An interesting result can be observed for VNS. Although this method used a good selection of LS operators, it obtained very poor results. This shows that this collection of LS operators can produce quite good results in certain cases, but this is rare. However, when these LS operators are used in the ILS-9 method, where they are paired with an IG operator and iterated to convergence, they can produce much better results. Thus, it can be seen that the strategy in which the LS operators are used has a significant impact on performance. GRASP achieved better results than any of the previously described methods. The result of SA shows that a simple method can already achieve quite good results. SA used the JS operator in combination with the RN perturbation operator, but achieved some of the best results of the existing metaheuristics. The fact that only a single LS operator was used could possibly be limiting, since the search is only performed in a single neighbourhood structure.

There are several reasons why the proposed ILS methods perform well compared to other metaheuristics. First, the method starts from a good initial solution, saving valuable computational time, but also focuses the search on a good solution region. Second, the method applies a carefully selected set of LS operators capable of covering the search space. Of course, this could be seen as a serious drawback of ILS methods, since there are many LS operators in the literature and therefore the decision is not trivial. However, the experiments have shown two things. First, even when all operators are used together, the method achieves quite good results and is among the best. Thus, it is not even necessary to make a detailed selection of the operators to obtain good results.

Second, the results of each LS operator have already identified several operators that do not perform well. These include the VIR and NSP perturbation operators, as well as most group-based operators and heuristic-based operators. Clearly, TD and JI have proven to be the best operators. Using just one of these operators yielded remarkable results, and using both together yielded some of the best results. This shows another extreme, namely that by using a small set of well-performing LS operators, the method again produces excellent results. So the designer has the choice of either selecting a few good LS operators or using a wide range of LS operators, and in both cases equally good results are obtained. Regardless of whether the LS operators are used to convergence or only once, the results show that both methods achieve similar performance.

The choices described in the previous paragraph are already sufficient to develop a method that works well. However, two additional improvements were considered. First, the PR was used to intensify the search. This method did not lead to any improvements and proved to be redundant. As mentioned earlier, this is likely due to the fact that the LS already traverse the search space quite extensively, making an intensification procedure redundant. On the other hand, the modified probabilistic solution acceptance criterion has proven to be much more useful. This acceptance criterion allows the method to start the search from another solution in the domain and thus escape from local optima more easily. The convergence plots showed that the methods reached good solutions quite quickly. Accepting worse solutions in the neighbourhood and starting the search from them allows the method to explore the search space in the area of good solutions more thoroughly.

From the results, it can be concluded that with careful design decisions, it is possible to construct a method that performs significantly better than more complicated metaheuristic methods. By generating a good initial solution, using a single operator to perturb the solutions, and only two LS operators, it is possible to obtain solutions that are less than 1% away from the lower bound. This shows that complicated and hybrid methods are not needed, but that even simple combinations of LS operators are more than sufficient to cope with the problem. Another aspect that must also be considered is that very often it is not even necessary to solve the problem optimally, but rather to a satisfactory degree. Therefore, one should not strive to define a complex and difficult-to-understand algorithm that will yield only a small increase in performance, but rather a method that is easy to understand and replicate, and that produces acceptable results.

6. Conlusion

This paper investigated the performance of various metaheuristics for the parallel unrelated machines scheduling problem. The goal was to find out whether it is necessary to use complicated metaheuristics for the above problem, or whether it is possible to obtain satisfactory results using simpler methods. The experiments show that simple methods based on LS operators are more powerful than some complex or hybrid methods. Metaheuristics such as GA,

ACO, and TS were not able to give as good results as those obtained by simple LS procedures. Although these metaheuristics could certainly be improved to match and perhaps eventually surpass the quality of the LS based methods, one should be cautious about concluding that such an approach is justified when acceptable solutions can already be obtained using simpler and more comprehensible methods. For example, a LS strategy that uses static ATC to initialise the solution, the IG perturbation operator to perturb the solutions, and only two LS operators may provide some of the best results obtained in this work. Each part is simple by itself and can be easily combined. On the other hand, more complicated metaheuristics must be adapted to such problems, which requires many design decisions that complicate the algorithm, or they are applied without much adaptation, which can usually lead to poor results.

Although this study has shown the superiority of several approaches over others, this does not mean that these methods are always superior to others. This depends heavily on the problem variant, optimisation criteria, and many other factors. Rather, the results should be interpreted in a way that very simple metaheuristic methods, coupled with good initial solutions, can significantly outperform the results of more complicated metaheuristics. Therefore, instead of developing new methods or highly complicated hybrid algorithms, one should first consider whether such problems can be satisfactorily solved with simpler methods and whether it is possible to improve performance by making better design decisions. More complex or hybrid methods should be considered if simpler methods do not provide acceptable results or cannot be adapted to some other constraints.

There are many plans to continue this research in the future. One of them is certainly to test the operators and methods to optimise other scheduling criteria and determine if there is a correlation between the considered criteria and the LS operators that should be applied. Another goal is to extend this research on multi-objective optimisation, especially to define simple LS based methods to optimise multiple criteria simultaneously. Another extension would be to incorporate various constraints into the problem, such as setup times, precedence constraints, machine eligibility, and the like. Finally, it would also be interesting to develop a method to automatically determine the set of LS operators that should be used. This could be done either offline before the schedule is created or even online during optimisation.

Declaration of interest and funding

This work has been supported in part by Croatian Science Foundation under the project IP-2019-04-4333.

References

Afzalirad, M., & Rezaeian, J. (2016). Design of high-performing hybrid metaheuristics for unrelated parallel machine scheduling with machine eligibility

and precedence constraints. *Engineering Optimization*, *48*, 706–726. doi:10.1080/0305215X.2015.1042475.

Al-Harkan, I., & Qamhan, A. (2019). Optimize unrelated parallel machines scheduling problems with multiple limited additional resources, sequence-dependent setup times and release date constraints. *IEEE Access*, *PP*, 1–1. doi:10.1109/ACCESS.2019.2955975.

Arık, O. (2019). Comparisons of metaheuristic algorithms for unrelated parallel machine weighted earliness/tardiness scheduling problems. *Evolutionary Intelligence*, . doi:10.1007/s12065-019-00305-7.

Arnaout, J.-P., Rabadi, G., & Musa, R. (2009). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *21*, 693–701. URL: https://doi.org/10.1007/s10845-009-0246-1. doi:10.1007/s10845-009-0246-1.

Balin, S. (2011). Non-identical parallel machine scheduling using genetic algorithm. *Expert Systems with Applications*, *38*, 6814–6821. URL: http://linkinghub.elsevier.com/retrieve/pii/S0957417410014272. doi:10.1016/j.eswa.2010.12.064.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, *6*, 154–160. doi:10.1287/ijoc.6.2.154.

Behnamian, J., Zandieh, M., & Fatemi Ghomi, S. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. *Expert Systems with Applications*, *36*, 9637–9644. URL: http://linkinghub.elsevier.com/retrieve/pii/S0957417408007252. doi:10.1016/j.eswa.2008.10.007.

Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., & Freund, R. F. (2001). A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing*, *61*, 810–837. URL: http://linkinghub.elsevier.com/retrieve/pii/S0743731500917143. doi:10.1006/jpdc.2000.1714.

Burke, E., De Causmaecker, P., Vanden Berghe, G., & Van Landeghem, H. (2004). The state of the art of nurse rostering. *J. Scheduling*, *7*, 441–499. doi:10.1023/B:JOSH.0000046076.75950.0b.

de C. M. Nogueira, J. P., Arroyo, J. E. C., Villadiego, H. M. M., & Goncalves, L. B. (2014). Hybrid GRASP Heuristics to Solve an Unrelated Parallel Machine Scheduling Problem with Earliness and Tardiness Penalties. *Electronic Notes in Theoretical Computer Science*, *302*, 53–72. URL: http:

//linkinghub.elsevier.com/retrieve/pii/S1571066114000218. doi:10.1016/j.entcs.2014.01.020.

Chang, P.-C., & Chen, S.-H. (2011). Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times. *Applied Soft Computing*, *11*, 1263 – 1274. URL: http://www.sciencedirect.com/science/article/pii/S1568494610000694. doi:https://doi.org/10.1016/j.asoc.2010.03.003.

Cheng, V., Crawford, L., & Menon, P. (1999). Air traffic control using genetic search techniques. In *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328)* (pp. 249–254). IEEE volume 1. URL: http://ieeexplore.ieee.org/document/806209/. doi:10.1109/CCA.1999.806209.

Chyu, C.-C., & Chang, W.-S. (2010). A competitive evolution strategy memetic algorithm for unrelated parallel machine scheduling to minimize total weighted tardiness and flow time. In *The 40th International Conference on Computers & Indutrial Engineering* (pp. 1–6). IEEE. URL: http://ieeexplore.ieee.org/document/5668388/. doi:10.1109/ICCIE.2010.5668388.

Costa, A., Cappadonna, F. A., & Fichera, S. (2013). A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, *69*, 2799–2817. URL: http://link.springer.com/10.1007/s00170-013-5221-5. doi:10.1007/s00170-013-5221-5.

Cota, L. P., Haddad, M. N., Souza, M. J. F., & Coelho, V. N. (2014). AIRP: A heuristic algorithm for solving the unrelated parallel machine scheduling problem. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1855–1862). IEEE. URL: http://ieeexplore.ieee.org/document/6900245/. doi:10.1109/CEC.2014.6900245.

Ezugwu, A. E. (2019). Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times. *Knowledge-Based Systems*, *172*, 15–32. URL: https://www.sciencedirect.com/science/article/pii/S0950705119300504. doi:https://doi.org/10.1016/j.knosys.2019.02.005.

Fanjul-Peyro, L., & Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, *207*, 55–69. URL: http://linkinghub.elsevier.com/retrieve/pii/S0377221710002572. doi:10.1016/j.ejor.2010.03.030.

Fanjul-Peyro, L., & Ruiz, R. (2011). Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research*, *38*, 301–309. URL: http://linkinghub.elsevier.com/retrieve/pii/S0305054810001188. doi:10.1016/j.cor.2010.05.005.

Fanjul-Peyro, L., & Ruiz, R. (2012). Scheduling unrelated parallel machines with optional machines and jobs selection. *Computers & Operations Research*, *39*, 1745–1753. URL: http://linkinghub.elsevier.com/retrieve/pii/S0305054811003005. doi:10.1016/j.cor.2011.10.012.

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research*, *101*, 173–182. URL: https://www.sciencedirect.com/science/article/pii/S0305054818301916. doi:https://doi.org/10.1016/j.cor.2018.07.007.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *J. Global Optimization*, *6*, 109–133.

Glass, C., Potts, C., & Shade, P. (1994). Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modelling*, *20*, 41–52. URL: https://doi.org/10.1016/0895-7177(94)90205-4. doi:10.1016/0895-7177(94)90205-4.

Glover, F. (1990). Tabu Search—Part II. *ORSA Journal on Computing*, *2*, 4–32. URL: http://pubsonline.informs.org/doi/abs/10.1287/ijoc.2.1.4. doi:10.1287/ijoc.2.1.4.

Glover, F., Laguna, M., & Marti, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, *29*.

Haddad, M. N., Coelho, I. M., Souza, M. J. F., Ochi, L. S., Santos, H. G., & Martins, A. X. (2012). GARP: A New Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem with Setup Times. In *2012 31st International Conference of the Chilean Computer Science Society* (pp. 152–160). IEEE. URL: http://ieeexplore.ieee.org/document/6694085/. doi:10.1109/SCCC.2012.25.

Hansen, J. V. (2004). Genetic search methods in air traffic control. *Computers & Operations Research*, *31*, 445–459. URL: http://linkinghub.elsevier.com/retrieve/pii/S0305054802002289. doi:10.1016/S0305-0548(02)00228-9.

Hansen, P., & Mladenovic, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, *130*, 449–467.

Hansen, P., Mladenovic, N., Todosijević, R., & Hanafi, S. (2016). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, *5*. doi:10.1007/s13675-016-0075-x.

Hart, E., Ross, P., & Corne, D. (2005). Evolutionary Scheduling: A Review. *Genetic Programming and Evolvable Machines*, *6*, 191–220. URL: http://link.springer.com/10.1007/s10710-005-7580-7. doi:10.1007/s10710-005-7580-7.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Complex adaptive systems (1st ed.). Cambridge, Mass: MIT Press.

Hop, N. V., & Nagarur, N. N. (2004). The scheduling problem of pcbs for multiple non-identical parallel machines. *European Journal of Operational Research*, *158*, 577 – 594. URL: http://www.sciencedirect.com/science/article/pii/S037722170300376X. doi:https://doi.org/10.1016/S0377-2217(03)00376-X.

Joo, C. M., & Kim, B. S. (2017). Rule-based meta-heuristics for integrated scheduling of unrelated parallel machines, batches, and heterogeneous delivery trucks. *Appl. Soft Comput.*, *53*, 457–476. URL: https://doi.org/10.1016/j.asoc.2016.12.038. doi:10.1016/j.asoc.2016.12.038.

Kim, C. O., & Shin, H. J. (2003). Scheduling jobs on parallel machines: a restricted tabu search approach. *The International Journal of Advanced Manufacturing Technology*, *22*, 278–287. URL: https://doi.org/10.1007/s00170-002-1472-2. doi:10.1007/s00170-002-1472-2.

Kim, D.-W., Kim, K.-H., Jang, W., & Chen, F. F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, *18*, 223 – 231. URL: http://www.sciencedirect.com/science/article/pii/S0736584502000133. doi:https://doi.org/10.1016/S0736-5845(02)00013-3. 11th International Conference on Flexible Automation and Intelligent Manufacturing.

Kim, D.-W., Na, D.-G., & Chen, F. F. (2003). Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer-Integrated Manufacturing*, *19*, 173 – 181. URL: http://www.sciencedirect.com/science/article/pii/S0736584502000777. doi:https://doi.org/10.1016/S0736-5845(02)00077-7. 12th International Conference on Flexible Automation and Intelligent Manufacturing.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*, 671–680. URL: http://www.sciencemag.org/cgi/doi/10.1126/science.220.4598.671. doi:10.1126/science.220.4598.671.

Kofler, M., Wagner, S., Beham, A., Kronberger, G., & Affenzeller, M. (2009). Priority Rule Generation with a Genetic Algorithm to Minimize Sequence Dependent Setup Costs. In R. Moreno-Díaz, F. Pichler, & A. Quesada-Arencibia (Eds.), *Computer Aided Systems Theory - EUROCAST 2009: 12th International Conference, Las Palmas de Gran Canaria, Spain, February 15-20, 2009, Revised Selected Papers* (pp. 817–824). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: http://link.springer.com/10.1007/978-3-642-04772-5_105. doi:10.1007/978-3-642-04772-5_105.

Lee, J.-H., Yu, J.-M., & Lee, D.-H. (2013). A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness. *The International Journal of Advanced Manufacturing Technology*, *69*, 2081–2089. URL: http://link.springer.com/10.1007/s00170-013-5192-6. doi:10.1007/s00170-013-5192-6.

Lee, Y. H., Bhaskaran, K., & Pinedo, M. (1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, *29*, 45–52. URL: http://www.tandfonline.com/doi/abs/10.1080/07408179708966311. doi:10.1080/07408179708966311.

Lenstra, J. K., Shmoys, D. B., & Tardos, E. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, *46*, 259–271. URL: http://link.springer.com/10.1007/BF01585745. doi:10.1007/BF01585745.

Lewis, R., Paechter, B., & Rossi-Doria, O. (2007). Metaheuristics for university course timetabling. In K. P. Dahal, K. C. Tan, & P. I. Cowling (Eds.), *Evolutionary Scheduling* (pp. 237–272). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-540-48584-1_9. doi:10.1007/978-3-540-48584-1_9.

Liao, T., Chang, P., Kuo, R., & Liao, C.-J. (2014). A comparison of five hybrid metaheuristic algorithms for unrelated parallel-machine scheduling and inbound trucks sequencing in multi-door cross docking systems. *Applied Soft Computing*, *21*, 180 – 193. URL: http://www.sciencedirect.com/science/article/pii/S1568494614001070. doi:https://doi.org/10.1016/j.asoc.2014.02.026.

Lin, C.-W., Lin, Y.-K., & Hsieh, H.-T. (2013). Ant colony optimization for unrelated parallel machine scheduling. *The International Journal of Advanced Manufacturing Technology*, *67*, 35–45. URL: http://link.springer.com/10.1007/s00170-013-4766-7. doi:10.1007/s00170-013-4766-7.

Logendran, R., McDonell, B., & Smucker, B. (2007). Scheduling unrelated parallel machines with sequence-dependent setups. *Computers & Operations Research*, *34*, 3420–3438. URL: http://linkinghub.elsevier.com/retrieve/pii/S0305054806000438. doi:10.1016/j.cor.2006.02.006.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). Iterated local search. In *Handbook of Metaheuristics* (pp. 320–353). Kluwer Academic Publishers. URL: https://doi.org/10.1007/0-306-48056-5_11. doi:10.1007/0-306-48056-5_11.

Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., & Freund, R. F. (1999). Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. *Journal of Parallel and Distributed Computing*, *59*, 107–131. URL: http://linkinghub.elsevier.com/retrieve/pii/S0743731599915812. doi:10.1006/jpdc.1999.1581.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, *24*, 1097 – 1100.

Molnar, G., Jakobović, D., & Pavelić, M. (2016). Workforce scheduling in inbound customer call centres with a case study. In G. Squillero, & P. Burelli (Eds.), *Applications of Evolutionary Computation* (pp. 831–846). Cham: Springer International Publishing.

Morton, T. E., & Pentico, D. W. (1993). *Heuristic Scheduling Systems*. John Wiley And Sons, Inc.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, *12*, 417–431. URL: http://link.springer.com/10.1007/s10951-008-0090-8. doi:10.1007/s10951-008-0090-8.

Petrovic, S., & Castro, E. (2011). A genetic algorithm for radiotherapy pretreatment scheduling. In C. Di Chio, A. Brabazon, G. A. Di Caro, R. Drechsler, M. Farooq, J. Grahl, G. Greenfield, C. Prins, J. Romero, G. Squillero, E. Tarantino, A. G. B. Tettamanzi, N. Urquhart, & A. Ş. Uyar (Eds.), *Applications of Evolutionary Computation: EvoApplications 2011: EvoCOMNET, EvoFIN, EvoHOT, EvoMUSART, EvoSTIM, and EvoTRANSLOG, Torino, Italy, April 27-29, 2011, Proceedings, Part II* (pp. 454–463). Berlin, Heidelberg: Springer Berlin Heidelberg.

Pinedo, M. L. (2012). *Scheduling: Theory, algorithms, and systems: Fourth edition* volume 9781461423614. Boston, MA: Springer US. URL: http://link.springer.com/10.1007/978-1-4614-2361-4. doi:10.1007/978-1-4614-2361-4. arXiv:arXiv:1011.1669v3.

Raja, K., Arumugam, C., & Selladurai, V. (2008). Non-identical parallel-machine scheduling using genetic algorithm and fuzzy logic approach. *International Journal of Services and Operations Management*, *4*, 72–101. doi:10.1504/IJSOM.2008.015941.

Rambod, M., & Rezaeian, J. (2014). Robust meta-heuristics implementation for unrelated parallel machines scheduling problem with rework processes and machine eligibility restrictions. *Computers and Industrial Engineering*, *77*, 15 – 28. URL: http://www.sciencedirect.com/science/article/pii/S0360835214002733. doi:https://doi.org/10.1016/j.cie.2014.09.006.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *177*, 2033–2049.

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, *22*, 3–18. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12001. doi:10.1111/itor.12001. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12001.

Srivastava, B. (1998). An effective heuristic for minimising makespan on unrelated parallel machines. *Journal of the Operational Research Society*, *49*, 886–894. doi:10.1057/palgrave.jors.2600547.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, *58*, 785 – 800. URL: http://www.sciencedirect.com/science/article/pii/S0360835210000483. doi:https://doi.org/10.1016/j.cie.2010.02.012.

Đurasević, M., & Jakobović, D. (2016). Comparison of solution representations for scheduling in the unrelated machines environment. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1336–1342). IEEE. URL: http://ieeexplore.ieee.org/document/7522347/. doi:10.1109/MIPRO.2016.7522347.

Đurasević, M., & Jakobović, D. (2018). A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications*, *113*, 555 – 569. URL: http://www.sciencedirect.com/science/article/pii/S0957417418304159. doi:https://doi.org/10.1016/j.eswa.2018.06.053.

Đurasević, M., & Jakobović, D. (2020). Comparison of schedule generation schemes for designing dispatching rules with genetic programming in the unrelated machines environment. *Applied Soft Computing*, *96*, 106637. URL: https://www.sciencedirect.com/science/article/pii/S1568494620305755. doi:https://doi.org/10.1016/j.asoc.2020.106637.

Đurasević, M., & Jakobović, D. (2018). Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. *Genetic Programming and Evolvable Machines*, *19*, 9–51. URL: https://doi.org/10.1007/s10710-017-9310-3. doi:10.1007/s10710-017-9310-3.

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, *211*, 612–622. URL: http://linkinghub.elsevier.com/retrieve/pii/S0377221711000142. doi:10.1016/j.ejor.2011.01.011.

Vlašić, I., Đurasević, M., & Jakobović, D. (2019). Improving genetic algorithm performance by population initialisation with dispatching rules. *Computers and Industrial Engineering*, *137*, 106030. URL: http://www.sciencedirect.com/science/article/pii/S0360835219304899. doi:https://doi.org/10.1016/j.cie.2019.106030.

Vlašić, I., Đurasević, M., & Jakobović, D. (2020). A comparative study of solution representations for the unrelated machines environment. *Computers & Operations Research*, *123*, 105005. URL: http:

//www.sciencedirect.com/science/article/pii/S0305054820301222. doi:https://doi.org/10.1016/j.cor.2020.105005.

Wotzlaw, A. (2012). *Scheduling Unrelated Parallel Machines: Algorithms, Complexity, and Performance*. AV Akademikerverlag.

Yang-Kuei, L., & Chi-Wei, L. (2013). Dispatching rules for unrelated parallel machine scheduling with release dates. *The International Journal of Advanced Manufacturing Technology*, *67*, 269–279. URL: http://link.springer.com/10.1007/s00170-013-4773-8. doi:10.1007/s00170-013-4773-8.