Undefined 0 (2023) 1–0
IOS Press

Constructing ensembles of dispatching rules for multi-objective tasks in the unrelated machines environment

Marko Đurasević [a,*], Francisco J. Gil-Gala [b], Domagoj Jakobović [a]
[a] *Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia*
[b] *Department of Computer Science, University of Oviedo, Spain*

Abstract. Scheduling is a frequently studied combinatorial optimisation that often needs to be solved under dynamic conditions and to optimise multiple criteria. The most commonly used method for solving dynamic problems are dispatching rules (DRs), simple constructive heuristics that build the schedule incrementally. Since it is difficult to design DRs manually, they are often created automatically using genetic programming. Although such rules work well, their performance is still limited and various methods, especially ensemble learning, are used to improve them. So far, ensembles have only been used in the context of single-objective scheduling problems This study aims to investigate the possibility of constructing ensembles of DRs for solving multi-objective (MO) scheduling problems. To this end, an existing ensemble construction method called SEC is adapted by extending it with non-dominated sorting to construct Pareto fronts of ensembles for a given MO problem. In addition, the algorithms NSGA-II and NSGA-III were adapted to construct ensembles and compared with the SEC method to demonstrate their effectiveness. All methods were evaluated on four MO problems with different number of criteria to be optimised. The results show that ensembles of DRs achieve better Pareto fronts compared to individual DRs. Moreover, the results show that SEC achieves equally good or even slightly better results than NSGA-II and NSGA-III when constructing ensembles, while it is simpler and slightly less computationally expensive. This shows the potential of using ensembles to increase the performance of individual DRs for MO problems.

Keywords: Unrelated machines environment, Genetic programming, Ensemble learning, Multi-objective optimisation

1. Introduction

Scheduling is a popular combinatorial optimisation problem where a certain number of jobs must be assigned to a finite number of machines, with the aim of optimising some user-defined criteria [1]. Due to their complexity, scheduling problems have usually been solved using various metaheuristic methods [2]. However, these methods can only be used to solve static scheduling problems where the information about all jobs is known in advance. On the other hand, they can hardly be applied to dynamic problems where not all properties of the problem are known a priori. For this reason, problem-specific heuristics, so-called dispatching rules (DRs), have been developed for solving dynamic scheduling problems.

DRs are simple heuristics that build the schedule incrementally by deciding which job to schedule next on a machine that becomes available [3]. As such, they only use the information currently available in the system to make the next immediate scheduling decision. Therefore, these rules can easily take into account any changes in the

*Corresponding author: Marko Đurasević, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, E-mail: marko.durasevic@fer.hr.

system, such as the arrival of new jobs, or the breakdown of a particular machine. Until recently, DRs were usually designed manually by domain experts. However, this trend is slowly changing towards the use of methods to automatically create new DRs. Of the many approaches used for this purpose, genetic programming (GP) [4] has been most commonly used to automatically design new DRs. GP has been used to design DRs for a variety of scheduling problems, such as the single machine environment [5], the job shop environment [6], unrelated parallel machines environment [7], the resource constrained project scheduling problem [8] and others [9]. Automatically designing DRs allows greater flexibility in their design as well as achieving DRs that perform better than existing manually designed DRs [7,10].

The added flexibility in automated DR design is important, especially when creating DRs for highly specific or more difficult problem variants, such as multi-objective (MO) scheduling problems. So far, most manually designed DRs have been created to optimise only a single criterion [11] However, the number of problems where multiple criteria need to be optimised is increasing, with various criteria like the makespan, total flowtime, total weighted tardiness, or total energy consumption being optimised simultaneously. Automatic design of DRs using GP solves this problem, as various MO optimisation algorithms can be coupled with GP to generate DRs that optimise multiple criteria simultaneously. Several studies have already shown that efficient MO DRs can be designed for various scheduling problems [12,13].

Even though automatically generated DRs outperform manually designed DRs, their performance is still limited, as it is difficult, if not impossible, to create a single rule that works well in all possible scheduling situations. Many methods have been described in the literature to improve the individual performance of DRs, such as using multitask learning [14] or surrogate models [15]. However, another line of research is the simultaneous use of multiple rules to perform scheduling decisions. Two of the most well-known approaches here are the selection of appropriate DRs based on certain characteristics of the problem instance [16] or the simultaneous use of multiple DRs to perform the scheduling decisions [17,18]. The latter method has gained attention especially in the context of automatically designed DRs of single-objective optimisation (SO). In such cases, the ensembles consisted of DRs designed for the optimisation of only one criterion, and as a result, the final ensemble was also specialised for the optimisation of only that specific objective.

However, ensembles can also be created from DRs developed for the simultaneous optimisation of several criteria. The question remains whether such a methodology is useful in the context of MO optimisation, as it is now important to obtain not only good but also robust solutions that generalise well. In a preliminary study, it was shown that an existing ensemble construction method can be adapted to construct ensembles from existing rules developed for multiple criteria optimisation [19]. Such ensembles were more efficient than single rules from which they were formed. Although the results are promising, the preliminary study only considered a single MO problem and did not perform a detailed analysis of the method. The aim of this study is to extend the previous one by introducing new ensemble design algorithms and investigating their performance on a wider set of MO problems. Furthermore, the results will be analysed in depth to gain better insight into the application of ensembles in the context of MO optimisation.

The remainder of the paper is arranged as follows. Section 2 contains the literature review dealing with automated design of DRs. Section 3 outlines the background of the study, more specifically the description of the unrelated machines environment and the automated design of DRs with GP. The methodology for creating ensembles of DRs for MO problems is described in Section 4. The experimental setup is described in Section 5, while the experimental results are presented in Section 6. Further analyses and discussion on the results are given in Section 7. Finally, Section 8 draws the conclusion of the paper and identifies potential research directions for the future.

2. Literature review

In recent years, GP has been increasingly used as a hyper-heuristic method [20], especially in the context of scheduling [9,21] and the uncapacitated arc routing problem [22,23]. Since the very first studies in the field of automated design of DRs, the field has evolved in many directions, including

surrogate models [15,24], feature selection [25,26], multitask GP [27,14], and others [28,16]. In addition to the aforementioned research directions, the design of MO DRs and the application of ensemble learning are two commonly explored areas.

2.1. Multi-objective optimisation

MO has become one of the most intensively researched areas of evolutionary computation, with applications in various fields such as transport [29,30], neuroevolution [31], computer animation [32], various engineering problems [33,34], machine learning [35] and feature selection [36]. In the context of automated DR design, Tay and Ho [37] were the first to consider developing DRs for a MO problem. They transformed the MO problem into a SO problem using a linear weighted combination of objectives. In [6], a MO algorithm was used for the first time to obtain Pareto fronts of DRs. The authors applied HaD-MOEA to optimise a single job shop scheduling problem with 5 criteria. The results obtained showed that DRs suitable for optimising multiple objectives can be developed. In [12], the authors extend the previous study by applying NSGA-II and NSGA-III to develop DRs for two MO scheduling problems.

Automatic design of MO DR for the unrelated machines environment was studied in [13], where the authors evaluated 4 MO algorithms on different problems with 3 to 9 criteria. The results show that automatically designed rules are better suited for MO optimisation than standard manually designed rules. In [38,39] the authors optimise a problem with multiple flowtime related objectives and use NSGA-II and SPEA2 to generate new DRs. In [40], the impact of different problem characteristics of the job shop environment on the performance of a MO GP method is analysed. The results show that the developed rules are robust enough to perform well on problems with different characteristics. In [41], the authors propose a novel GP method coupled with a Pareto local search to improve the quality of the developed DRs. Finally, in [42], simultaneous optimisation of energy efficiency and mean delay is considered, but the focus of the study is on schedule construction rather than MO optimisation.

2.2. Ensemble learning

Ensemble learning is a popular machine learning method for building powerful classification and prediction models [43], which have applications in various fields, such as predicting the cracking risk of concrete, emotions [44] or deception detection [45]. Especially recently, ensembles in deep learning [46,47] have attracted much attention, where single they obtain better generalisation properties for various problems such as facial expression recognition [48] or object recognition [49].

The first application of ensembles for DRs was in [17], where cooperative coevolution was used to evolve ensembles of DRs. In this method, the entire population was divided into a fixed number of subpopulations, with each subpopulation evolving one rule of the ensemble. The experiments showed that ensembles created in this way performed better than individual DRs. Therefore, the research was extended in [50], by using a GP method that developed groups of rules (ensembles) in the first stage and the individual rules in the second stage. However, no significant improvement in the results was achieved compared to the basic method of [17]. In [51], a novel method for creating ensembles of DRs was proposed, NELLI-GP, where rules are developed for subsets of problem instances to specialise in, and then combined into ensembles. The results obtained show an improvement over previous methods.

In [52], the authors compare different methods to combine the decisions of each DR in the ensembles to reach the final ensemble decision. Standard ensemble combination methods from machine learning were used, such as voting, weighted voting, sum and weighted sum. The results show that the best ensembles were obtained using the sum combination method. Four ensemble construction methods, cooperative coevolution, BagGP, BoostGP and SEC, were studied in [53]. Cooperative coevolution is similar to the method used in [17], while BagGP and BoostGP are GP variants inspired by bagging and boosting methods from machine learning. SEC is a novel method that uses a different intuition when constructing ensembles. Instead of developing DRs at the same time as constructing ensembles, SEC assumes that already developed DRs exist and uses them to construct ensembles by randomly sampling these rules. Results show that SEC performs

significantly better than alternative methods that develop DRs and construct ensembles simultaneously. The performance of SEC is validated in [54] by applying it to the problem of resource constrained project scheduling and in [18] by a deeper analysis of the method.

In [5], the authors propose a new type of ensembles, where each rule is executed independently to create the entire schedule, and the best solutions obtained are selected. These ensembles were further investigated in [55], where they were constructed from the manually designed ATC rules, which did not show as good results as when using GP evolved rules. In [56], the authors also investigated the performance of different methods for constructing ensembles and found that the proposed memetic algorithm performed the best. Since the previous type of ensembles constructs the entire schedule, it is only applicable to static problems where all information about the system is known in advance. To alleviate this problem, in[57] this ensemble method is modified not to construct the entire schedule, but only a portion of the schedule based on the information known up to that point. With this modification, the ensembles can be used for dynamic environments and they outperform the standard ensemble variants that use the sum or vote combination method.

2.3. Ensemble learning for MO problems

To date, only one preliminary study has examined the use of ensembles of DRs in the context of solving MO scheduling problems [19]. This study considered a single MO problem involving three criteria and reported a brief analysis of the results. The results were promising and indicated that ensembles of DRs could be used in the context of MO optimisation to obtain better results than by individual DRs. This motivated us to extend this research and conduct a more comprehensive investigation of the application of ensembles to solve MO scheduling problems.

3. Background

3.1. The unrelated machines environment

The unrelated machines environment is a scheduling problem in which a set of n jobs have to be scheduled on one of the m available machines [1]. Each job in the environment has the following properties:

- processing time p_{ij} – time required to process job j on machine i.
- release time r_j – arrival time of job j into the system.
- due date d_j – time until job j should finish executing.
- weight w_j – importance of job j.

Using the properties described earlier, a schedule can be constructed for this problem. Based on the schedule created, additional properties can be calculated for each job, which include:

- Completion time C_j – the time when job j completed with its execution.
- Flowtime F_j – the amount of time that the job spent in the system, defined as $F_j = C_j - r_j$.
- Tardiness flag U_j – denotes whether job j was tardy or not. If job j was tardy then $U_j = 1$, otherwise $U_j = 0$.
- Tardiness T_j – the amount of time that job j was tardy, defined as $T_j = \max(C_j - d_j, 0)$.

Based on the previously defined job properties several criteria can be used to evaluate the quality of schedules. This study considers the following:

- C_{max} - maximum completion time of all jobs: $C_{max} = \max_j(C_j)$.
- Ft - total flowtime: $Ft = \sum_j F_j$.
- Mus - machine usability: $M_{ut} = \max_i(\frac{P_i}{C_{max}}) - \min_i(\frac{P_i}{C_{max}})$, where P_i is defined as the sum of processing times of all jobs executed on machine i.
- Nwt - weighted number of tardy jobs: $Nwt = \sum_j w_j U_j$.
- Twt - total weighted tardiness: $Twt = \sum_j w_j T_j$.

The C_{max}, Ft, Nwt, and Twt criteria were selected because they are among the most frequently considered criteria in the unrelated machines environment[11], while Mus was selected because optimising this criterion negatively affects other scheduling criteria.

Although the problem described earlier can be considered under both static and dynamic conditions, this study examines the problem under dynamic conditions where no information about a job is known until it is released in the system. Once

it is released, all information about it is available and it can be immediately considered for scheduling. Therefore, standard search based methods are not applicable to this problem variant and the focus has to be shifted to simple constructive heuristics, namely DRs.

3.2. Construction of MO DRs

As mentioned earlier, DRs build the solution to a scheduling problem incrementally by deciding at each decision point which job to schedule on which machine. This makes DRs suitable for solving dynamic scheduling problems [3]. They can be divided into a schedule generation scheme and a priority function.

The schedule generation scheme (SGS) represents the outline of the DR, i.e. it determines when a scheduling decision must be made and which decisions are eligible for it. Each time the SGS determines that there is at least one released job and one available machine, it determines whether to schedule one of the released jobs on one of the available machines. To make this decision, it uses a priority function (PF) to calculate the priorities for assigning each of the available jobs to each of the machines. Based on the priority values obtained for each decision, the SGS selects the job with the best priority and schedules it. Due to its simplicity, the SGS is designed manually as it usually follows a similar pattern, although some design decisions may affect its performance [58].

The PF used by the SGS uses certain system properties to calculate a numerical value for each possible decision. For example, a simple PF could schedule jobs according to their release times, i.e. jobs would be scheduled on a "first in, first out" (FIFO) basis. Unfortunately, designing good PFs for various criteria and scheduling problems is a tedious task, which is why this process is usually automated using various methods, most notably genetic programming GP and similar methods [59] The reason for this is that PFs can be easily represented by expression trees, which is exactly the representation used by GP.

To apply GP to generate PFs, it is necessary to define the primitive set of nodes that will be used to construct the PFs. Table 1 shows the set of terminal and function nodes used by GP, selected based on preliminary experiments [7]. The terminal nodes provide basic information about the problem such as the processing time (pt), but also more complex information such as the time remaining before a job is late (SL). The set of function nodes includes only the most basic function nodes, as the inclusion of more complex functions did not improve the results [7].

Table 1
The symbols set

Node	Description
	Terminals
pt	processing time of job j on machine i
$pmin$	minimal processing time (MPT) of job j
$pavg$	average processing time of job j across all machines
PAT	time until machine with the MPT for job j becomes available
MR	time until machine i becomes available
age	time which job j spent in the system
dd	time until which job j has to finish with its execution
w	weight of job j (w_j)
SL	slack of job j, $-max(d_j - p_{ij} - t, 0)$
	Functions
$+$	binary addition
$-$	binary subtraction
$*$	binary multiplication
$/$	binary secure division, returns 1 if division by 0 occurs
POS	$POS(x) = \max(x, 0)$

Since the standard GP is only suitable for optimising a single objective, it must be combined with a MO algorithm to optimise MO problems. In this study, GP is used in combination with the NSGA-II [60] and NSGA-III [61] algorithms to develop DRs suitable for optimising multiple objectives simultaneously.

4. Constructing ensembles for MO problems

4.1. Ensemble Construction

In order to use ensembles for a given problem, the methods for both constructing and combining ensembles need to be defined. The *ensemble construction* method specifies how ensembles are constructed from individual DRs. The construction of MO ensembles is outlined in Figure 1 and can be divided into three phases. In the first phase, GP is used in conjunction with the MO algorithms, NSGA-II and NSGA-III, to evolve Pareto fronts of DRs for a given problem, where a given set of problem instances for training. The GP can be run

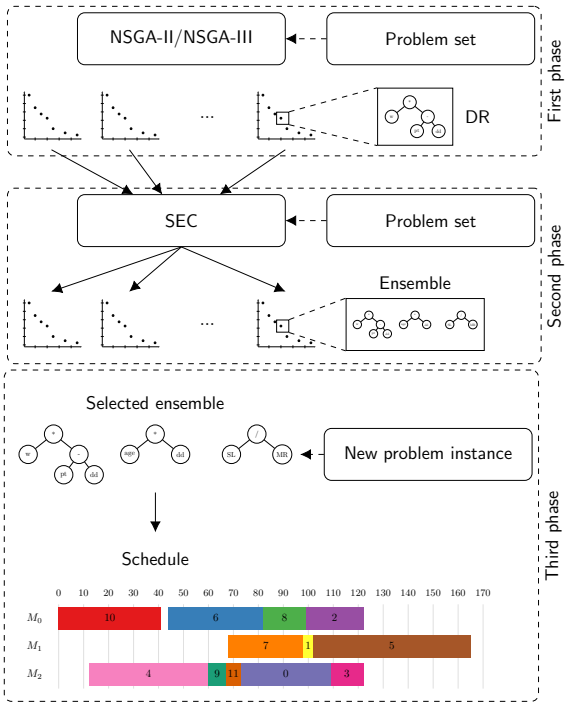


Fig. 1. Outline of the proposed method

any number of times, and each time it is run, the final Pareto front of DRs is obtained as a result.

In the second phase, the DRs obtained from the first phase are used by an ensemble construction method to create ensembles. For ensemble construction, the previously proposed SEC method for SO problems is used [53]. Although different strategies can be used to select the DRs that will form the ensemble [18], the original version randomly selects K DRs from the set of available DRs, where K represents the size of the ensemble to be constructed. This process is repeated N times and the best ensemble is returned as the solution.

However, the original SEC method is not applicable to MO problems, since a Pareto set of solutions must be determined instead of a single solution. Therefore, SEC is modified as described in Algorithm 1 by introducing the concept of non-dominated sorting. The method has three parameters: the number of ensembles it constructs N, the size of the ensemble ES, and the set of rules used to construct the ensembles R. In each iteration, the method constructs an ensemble by randomly selecting ES rules from R and stores it in a set of constructed ensembles. When the required number of ensembles is constructed, the set of ensembles is sorted non-dominantly and divided into fronts. The first Pareto front, consisting of solutions that are not dominated by any other solution, is then returned as the final result. The method can be executed once or several times. The result of the second phase are the Pareto fronts obtained from each execution.

Instead of using SEC to create ensembles, other methods can also be used in the second phase. Therefore, NSGA-II and NSGA-III are used instead of SEC to construct ensembles from existing DRs and to assess the performance of the SEC method. Both algorithms use a simple integer representation, where each gene represents the index of the DR to be included in the ensemble. The algorithm uses a crossover operator that uniformly selects DRs from each parent, and a mutation that randomly changes one DR in the individual to another DR. These variants of the MO algorithms used to create the ensembles will be referred to as E-NSGA-II and E-NSGA-III in the rest of the text.

Furthermore, during the construction of ensembles, two execution strategies can be used to determine which of the DRs obtained in the first phase should be used in the construction of ensembles in the second phase. The first strategy, referred to as IND, uses only DRs obtained from a single GP run in the first phase to construct ensembles in the second phase. This strategy is less computationally intensive as only a single GP run needs to be performed in the first phase, although it results in a less diverse set of DRs. In the second strategy, referred to as ALL, the Pareto sets of DRs obtained from multiple GP executions in the first phase are merged and used to form ensembles. This strategy is more computationally intensive as multiple executions of GP must be performed in the first phase, although the individual runs can be performed in parallel. However, this strategy results in more diverse sets of DRs that are used to form ensembles.

Finally, in the third phase, when a new scheduling problem needs to be solved, an ensemble is selected based on the trade-off between the criteria and used to create the schedule for the problem at hand. It is important to note that the first two phases can be performed at any time before a scheduling problem is solved, so both DRs and ensembles are created in advance. This means that when a new problem instance is solved, one of the

Algorithm 1 The adapted SEC method for MO problems

Data: N, ES, R
Result: The first front of solutions
while $|$ *constructed* $|< N$ **do**
  $E \leftarrow \emptyset$;
  while $|\ E\ | < ES$ **do**
    Select a random DR from $R \setminus E$, and add it to E;
  end
  constructed $\leftarrow$ *constructed* $\cup \{E\}$;
end
Perform nondominated sorting on the *constructed* set;
return *first front of constructed*

previously developed ensembles can be used immediately to create the schedule, which significantly reduces the complexity of the approach and makes it suitable for dynamic real-time environments.

4.2. Ensemble Combination

In order for the ensemble to come to a decision, two *ensemble combination* methods are used, sum and vote [52]. In the sum combination method, the priorities of all rules are summed to a single value, and the scheduling decision with the lowest aggregated value is selected and executed. In the voting combination method, on the other hand, each rule casts a vote for a single scheduling decision (the one that received the lowest priority value) and the decision with the most votes is executed. In the event of a tie, the job that was released the soonest is selected and executed. A potential problem in the sum combination method is that a single DR could completely control the behaviour of the ensemble if its priorities are of larger magnitudes than those of other rules. This problem does not appear in the vote method, since the decision of each rule is equally important, but ties are more likely to occur especially for smaller ensembles.

An example of how these two methods perform the decision for an ensemble of three rules is shown in Figure 2. In this example, the ensemble must decide which of three available jobs to schedule next. Therefore, each rule in the ensemble calculates the priority for each job, denoted as π_j, based on which the ensemble's combination methods make their decision. As can be seen, the combination methods can arrive at different decisions even though they consist of the same DRs.

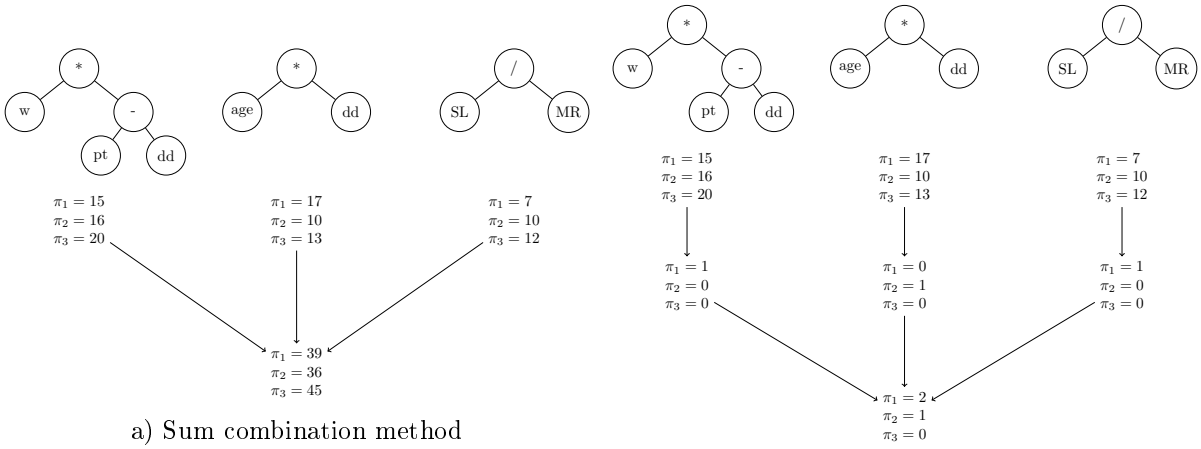
5. Experimental setup

This section explains the design of the experiments. In the first phase, DRs were developed for the MO problems considered by coupling GP with NSGA-II and NSGA-III. The reason why NSGA-II and NSGA-III were chosen to perform MO optimisation is that previous studies have shown that they generally perform better than other MO methods such as MOEA/D or SPEA2 [13,12]. For both algorithms, a population consisting of 1 000 individuals was used, while the mutation probability was set to 0.1 for NSGA-II and 0.9 for NSGA-III. The termination criterion of 100 000 function evaluations was used. In addition, the subtree, uniform, context-preserving and size-fair crossover operators were used, as well as the subtree, hoist, node complement, node replacement, permutation and shrink mutation operators [4]. All parameters were selected based on previous experiments [13]. Both algorithms were run 30 times, and the result of each run is a Pareto front of non-dominated DRs.

The DRs from phase 1 are used by the ensemble construction methods to create ensembles of DRs in the second phase using SEC, E-NSGA-II, and E-NSGA-III. E-NSGA-II and E-NSGA-III use the same parameters as NSGA-II and NSGA-III to develop individual rules. All algorithms use a termination criterion of 10 000 function evaluations of ensembles and are tested with ensemble sizes of 3, 5 and 7 and with the sum and vote combination methods. All experiments were run 30 times to obtain statistically significant results.

To develop and test the rules and ensembles, a set of 120 instances from previous studies was used [7], in which the number of jobs and machines ranged from 12 to 100 and from 3 to 10, respectively. These instances were divided into two independent sets, the training set and the test set. The training set is used by GP to evolve individual DRs and by the ensemble construction methods to form ensembles. The test set, on the other hand, is used to evaluate the quality of the obtained Pareto fronts of rules and ensembles in unseen problems.

All methods were evaluated on four selected MO unrelated machines scheduling problems. Using the standard notation for scheduling problems [1], each problem is represented as $R \mid r_j \mid X$, where R denotes the unrelated machines environment, r_j indicates that job release times are con-



a) Sum combination method

b) Vote combination method

Fig. 2. Example of the ensemble combination methods.

sidered, and X represents the set of optimised criteria. Specifically, the following optimisation problems are considered:

- $R \mid r_j \mid C_{max}, Twt$
- $R \mid r_j \mid C_{max}, Mus, Twt$
- $R \mid r_j \mid C_{max}, Ft, Mus, Twt$
- $R \mid r_j \mid C_{max}, Ft, Mus, Nwt, Twt$

More MO problems were assessed, but it was found that for the same number of objectives the methods produced comparable results and similar conclusions could be drawn. Therefore, for each number of objectives, a single problem was selected to outline and analyse the results.

Since the quality of the Pareto fronts is being examined, the hypervolume (HV) indicator is used to evaluate the Pareto fronts obtained [62]. The reason for choosing this indicator is that it is the most widely used [63], but also because other studies have confirmed its efficiency compared to other indicators [64]. For each problem, there is a different reference point, which is obtained by selecting the worst values for each criterion. In addition, the HV values are normalised to scale them between 0 and 1. Since 30 runs were made for each method, the tables in the results section only give the median of the HV values obtained for each experiment.

In a first analysis, all data obtained were checked with the Shapiro-Wilk test to see if they were normally distributed. Since the results of most experiments were not normally distributed, the statistical differences between the results were tested with the non-parametric Kruskal-Wallis test and the Dunn post hoc test to determine the pairwise differences with the Bonferroni correction method. The differences are considered significant if a p-value below 0.05 is obtained, as this value is commonly used in other similar studies to infer the significance of the results [10,14].

All algorithms and experiments were implemented using in the C++ programming language using the ECF[1] framework. All experiments were executed on a system with AMD Ryzen Threadripper 3990X 64-Core Processor 2.90 GHz, 128 GB RAM, and the Windows 10 operating system.

6. Results

6.1. Optimisation of two criteria

Table 2 shows the results obtained for the $R \mid r_j \mid C_{max}, Twt$ problem. The first part of the table (labelled 'IND' in the rule rules column) describes the results of ensembles created with DRs from individual Pareto fronts, i.e. the results obtained from a single MO algorithm run. Then, the results for ensembles created with the union of all Pareto fronts are presented (labelled 'ALL'). For each experiment, the algorithm used to develop each rule is given in the 'RCM' column, while the method used to create the ensemble is given in the 'ECM' column. Finally, at the bottom the results

[1] http://ecf.zemris.fer.hr/

for the individual rules developed by NSGA-II and NSGA-III are outlined. The results of the statistical tests when comparing ensembles and individual rules are given in brackets next to the HV values. The first entry in the brackets denotes the result of the comparison between the ensemble and the rules developed by NSGA-II, while the second entry denotes the result of the comparison between the ensembles and the rules obtained from NSGA-III. The symbol + means that the ensembles perform significantly better than the individual DRs, $\approx$ that they perform equally well, and $-$ that the ensembles perform significantly worse than the individual rules.

The results for this problem show that no improvement can be achieved when ensembles are created from individuals obtained from a single run in the first phase of the algorithm. However, even in the ALL scenario, when multiple Pareto fronts of DRs are combined and used to create ensembles, the ensembles only achieve better performance than DRs in a few cases. SEC and the two GA methods used to create ensembles (E-NSGA-II, E-NSGA-III) achieve similar performance with no significant differences. As for the ensemble parameters, slightly better HV values are obtained when the vote combination method and ensemble sizes 3 and 5 are used. Finally, we find that all ensembles created from the rules developed by NSGA-II in the ALL scenario perform better than individual rules, for all parameter values.

The results for the ALL scenario are shown in Figure 3. Here we see that ensembles created using DRs developed by NSGA-II lead to improvements in results that are less dispersed than the results of individual DRs. However, the same is not true for ensembles constructed from rules developed by NSGA-III, as they perform as well as individual rules. The reason for this is not entirely clear, as no significant differences were found when examining the Pareto fronts obtained with both NSGA-II and NSGA-III. Finally, we see that smaller ensembles tend to perform better with the vote combination, although this does not occur consistently in all experiments to be accepted as a general rule.

To illustrate the Pareto fronts, they are shown for DRs (developed by NSGA-III) and ensembles (constructed with SEC using rules developed by NSGA-III for the best combination of parameters) in Figure 4. The figure shows the total Pareto fronts merged from all algorithm executions. We

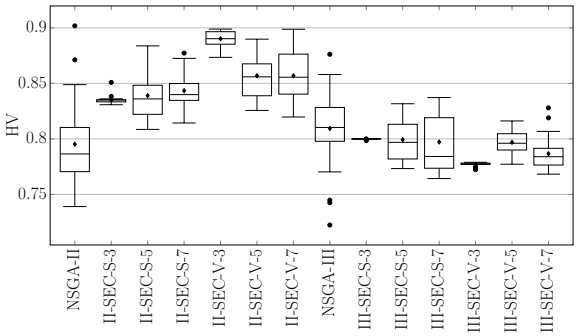


Fig. 3. Box plot of the results for the $R \mid r_j \mid C_{max}, Twt$ problem.

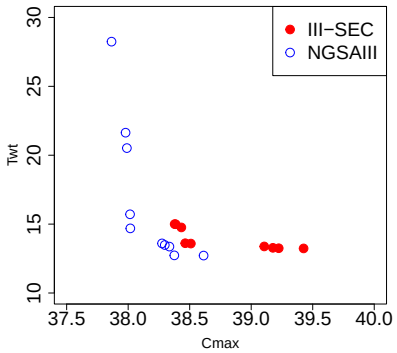


Fig. 4. The Pareto front obtained for $R \mid r_j \mid C_{max}, Twt$ problem.

see that the Pareto front obtained by ensembles are inferior to those obtained by individual rules, giving better coverage of the objective space as well as better convergence of solutions. The likely reason for this is that SEC had only a limited number of DRs (36) available for creating ensembles. This number of rules seems to be insufficient and the ensemble construction methods were not able to construct ensembles that outperform individual DRs. This suggests that a larger set of ensemble construction rules is needed for MO problems, which will be examined in the following sections.

6.2. Optimisation of three criteria

Table 3 gives an overview of the results obtained when optimising the $R \mid r_j \mid C_{max}, Mus, Twt$ problem. We see that forming ensembles from individual Pareto fronts does not yield significant improvement. However, the ensembles created from the combined Pareto fronts perform well overall,

Table 2
Results for the HV metric for the $R \mid r_j \mid C_{max}, Twt$ problem

Rules	RCM	ECM	sum			vote		
			3	5	7	3	5	7
IND	NSGA-II	E-NSGA-II	0.77 ($\approx$|$\approx$)	0.77 ($\approx$|$\approx$)	0.76 ($\approx$| $-$)	0.79 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)	0.77 ($\approx$|$\approx$)
	NSGA-III	E-NSGA-III	0.79 ($\approx$|$\approx$)	0.79 ($\approx$|$\approx$)	0.77 ($\approx$|$\approx$)	0.80 ($\approx$|$\approx$)	0.79 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)
	NSGA-II	SEC	0.77 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)	0.76 ($\approx$|$\approx$)	0.79 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)
	NSGA-III	SEC	0.79 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)	0.80 ($\approx$|$\approx$)	0.80 ($\approx$|$\approx$)	0.79 ($\approx$|$\approx$)
ALL	NSGA-II	E-NSGA-II	0.84 ($+$|$\approx$)	0.85 ($+$|$\approx$)	0.83 ($+$|$\approx$)	0.88 ($+$|$+$)	0.85 ($+$|$\approx$)	0.85 ($+$|$\approx$)
	NSGA-III	E-NSGA-III	0.80 ($\approx$|$\approx$)	0.80 ($\approx$|$\approx$)	0.77 ($\approx$| $-$)	0.78 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)	0.80 ($\approx$|$\approx$)
	NSGA-II	SEC	0.84 ($+$|$\approx$)	0.85 ($+$|$\approx$)	0.83 ($+$|$\approx$)	0.88 ($+$|$+$)	0.85 ($+$|$\approx$)	0.85 ($+$|$\approx$)
	NSGA-III	SEC	0.80 ($\approx$|$\approx$)	0.80 ($\approx$|$\approx$)	0.79 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)	0.80 ($\approx$|$\approx$)	0.78 ($\approx$|$\approx$)
NSGA-II			0.79					
NSGA-III			0.81					

as can be seen from the ensembles created by SEC, where all ensembles performed significantly better than individual DRs. On the other hand, the ensembles created by GAs were not as successful and did not achieve better results for all parameter combinations. However, there were no significant differences between the ensembles created by SEC or the two GAs, which means that both methods construct ensembles of similar quality.

From Figure 5 we see that, compared to individual DRs, ensembles not only produce much better results, but also less scattered results. Therefore, in this problem, it was possible to significantly improve the results by using ensembles. As far as the influence of the parameters is concerned, the sum combination method performs equally well regardless of the size of the ensemble. This cannot be said for the vote combination method, as it performs best when only three rules are used in the ensemble. Furthermore, we see that at ensemble size 3, the vote combination method performs better than the sum method, which changes as the ensemble size increases and the sum combination method performs better. This is an interesting behaviour, suggesting that it is more difficult to create larger ensembles of good quality using the vote combination method. The reasons for this are discussed in a later section.

Figure 6 outlines the Pareto fronts obtained by MO DRs and ensembles. Since it is difficult to visualise the Pareto fronts for three criteria, the Pareto fronts are given for each pairwise combination of criteria. The figure shows that ensembles achieve better coverage of the objective space, but also better convergence, which is best seen in from Figures 6a and 6c. In these figures we see that the ensembles in the middle of the objective space perform better than individual DRs. This means that ensembles are much better than DRs at finding solutions that offer a trade-off between the optimised criteria, which is important in MO optimisation. On the other hand, ensembles and rules perform equally well at the extremes of the objectives, i.e. when they focus more on optimising a single objective, they perform equally well.

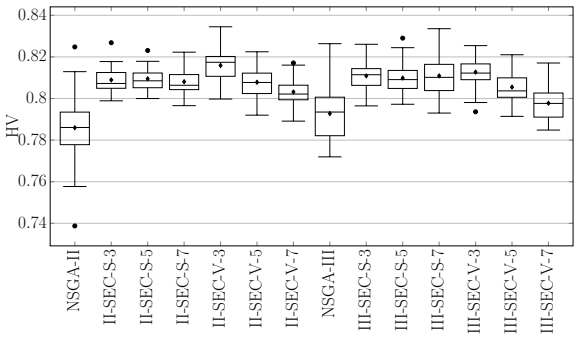


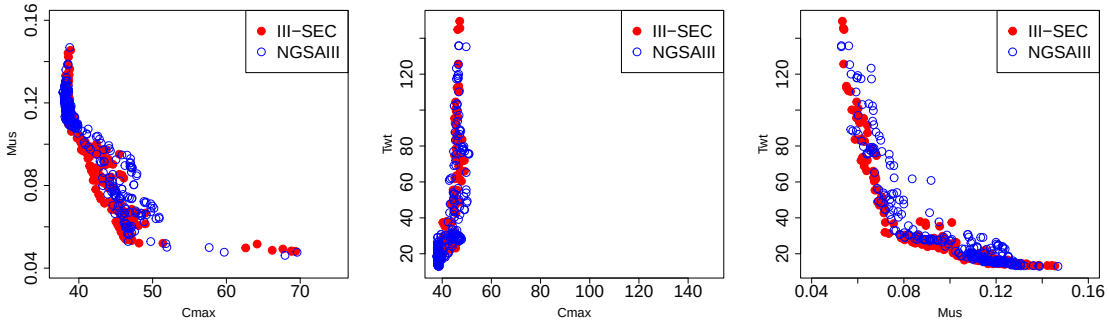
Fig. 5. Box plot of the results for the $R \mid r_j \mid C_{max}, Mus, Twt$ problem.

6.3. Optimisation of four criteria

The results for the $R \mid r_j \mid C_{max}, Ft, Mus, Twt$ problem are given in Table 4. Again, ensembles

Table 3
Results for the HV metric for the $R \mid r_j \mid C_{max}, Mus, Twt$ problem

Rules	RCM	ECM	sum			vote		
			3	5	7	3	5	7
IND	NSGA-II	E-NSGA-II	0.78 (≈|≈)	0.77 (≈|≈)	0.77 (≈|≈)	0.78 (≈|≈)	0.78 (≈|≈)	0.73 (≈|≈)
	NSGA-III	E-NSGA-III	0.79 (≈|≈)	0.78 (≈|≈)	0.77 (≈|≈)	0.80 (≈|≈)	0.79 (≈|≈)	0.78 (≈|≈)
	NSGA-II	SEC	0.79 (≈|≈)	0.78 (≈|≈)	0.77 (≈|≈)	0.79 (≈|≈)	0.78 (≈|≈)	0.78 (≈|≈)
	NSGA-III	SEC	0.79 (≈|≈)	0.78 (≈|≈)	0.77 (≈|≈)	0.79 (≈|≈)	0.79 (≈|≈)	0.78 (≈|≈)
ALL	NSGA-II	E-NSGA-II	0.81 (+ |≈)	0.81 (+ | +)	0.81 (+ |≈)	0.81 (+ | +)	0.80 (≈|≈)	0.80 (≈|≈)
	NSGA-III	E-NSGA-III	0.81 (+ |≈)	0.80 (≈|≈)	0.81 (+ | +)	0.81 (≈|≈)	0.80 (≈|≈)	0.79 (≈|≈)
	NSGA-II	SEC	0.81 (+ | +)	0.81 (+ | +)	0.81 (+ | +)	0.82 (+ | +)	0.81 (+ | +)	0.80 (≈|≈)
	NSGA-III	SEC	0.81 (+ | +)	0.81 (+ | +)	0.81 (+ | +)	0.81 (+ | +)	0.81 (+ |≈)	0.80 (≈|≈)
NSGA-II			0.79					
NSGA-III			0.79					



a) C_{max} & Mus b) C_{max} & Twt c) Mus & Twt

Fig. 6. The Pareto front obtained for the $R \mid r_j \mid C_{max}, Mus, Twt$ problem denoted through pairwise combinations of the three optimised criteria Pareto front.

created from individual Pareto fronts do not perform well. However, those created from the union of all Pareto fronts perform much better, especially if they use rules developed by NSGA-II. In this case, the ensembles created by SEC perform significantly better than individual DRs in all but one experiment. The SEC method performs better than both GAs, as it significantly outperforms the results of the individual DRs in more cases.

Figure 7 outlines the boxplot of the results. Compared to individual DRs, ensembles achieve better distributed results in most of the experiments performed. However, as in the previous problem, we observe some deterioration in the quality of the results when the size of the ensemble is increased. And while this was only observed for the vote combination method in the last problem, such behaviour is observed here for both methods, although it is more pronounced for the vote combination method. Thus, for the largest ensemble sizes, the results are similar to those obtained with single DRs. Furthermore, the sum combination method achieves better results, since in the case of using DRs developed by NSGA-II, it clearly outperforms the vote combination methods for ensemble sizes 5 and 7.

Figure 8 shows the obtained Pareto fronts, which show that ensembles provide better coverage of the objective space compared to individual DRs. For pairs of criteria that are not conflicting, such as those shown in Figures 8a, 8c and 8e, both methods obtain results that optimise both criteria well. However, when considering the Pareto fronts including the Mus criterion, which conflicts with

Table 4
Results for the HV metric for the $R \mid r_j \mid C_{max}, Ft, Mus, Twt$ problem

Rules	RCM	ECM	sum			vote		
			3	5	7	3	5	7
IND	NSGA-II	E-NSGA-II	0.75 (≈|≈)	0.74 (≈|≈)	0.73 (≈|≈)	0.75 (≈|≈)	0.74 (≈|≈)	0.74 (≈|≈)
	NSGA-III	E-NSGA-III	0.74 (≈|≈)	0.74 (≈|≈)	0.73 (≈|≈)	0.75 (≈|≈)	0.74 (≈|≈)	0.73 (≈|≈)
	NSGA-II	SEC	0.75 (≈|≈)	0.74 (≈|≈)	0.74 (≈|≈)	0.76 (≈|≈)	0.75 (≈|≈)	0.75 (≈|≈)
	NSGA-III	SEC	0.75 (≈|≈)	0.74 (≈|≈)	0.73 (≈|≈)	0.75 (≈|≈)	0.75 (≈|≈)	0.73 (≈|≈)
ALL	NSGA-II	E-NSGA-II	0.77 (+ | +)	0.77 (≈|≈)	0.76 (≈|≈)	0.77 (+ | +)	0.76 (≈|≈)	0.75 (≈|≈)
	NSGA-III	E-NSGA-III	0.78 (+ | +)	0.76 (≈|≈)	0.75 (≈|≈)	0.77 (+ | +)	0.76 (≈|≈)	0.76 (≈|≈)
	NSGA-II	SEC	0.78 (+ | +)	0.77 (+ | +)	0.77 (+ | +)	0.78 (+ | +)	0.77 (+ | +)	0.75 (≈|≈)
	NSGA-III	SEC	0.77 (+ | +)	0.76 (≈|≈)	0.75 (≈|≈)	0.77 (+ | +)	0.76 (≈|≈)	0.75 (≈|≈)
NSGA-II			0.75					
NSGA-III			0.75					

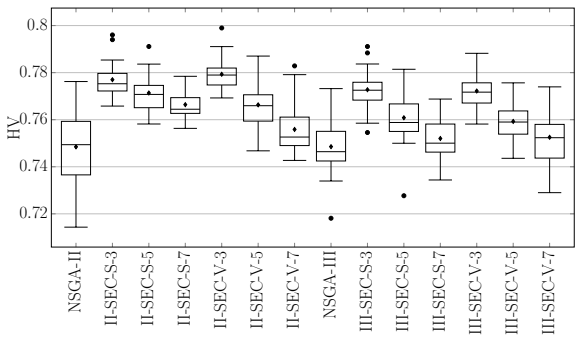


Fig. 7. Results for the HV metric for the $R \mid r_j \mid C_{max}, Ft, Mus, Twt$ problem.

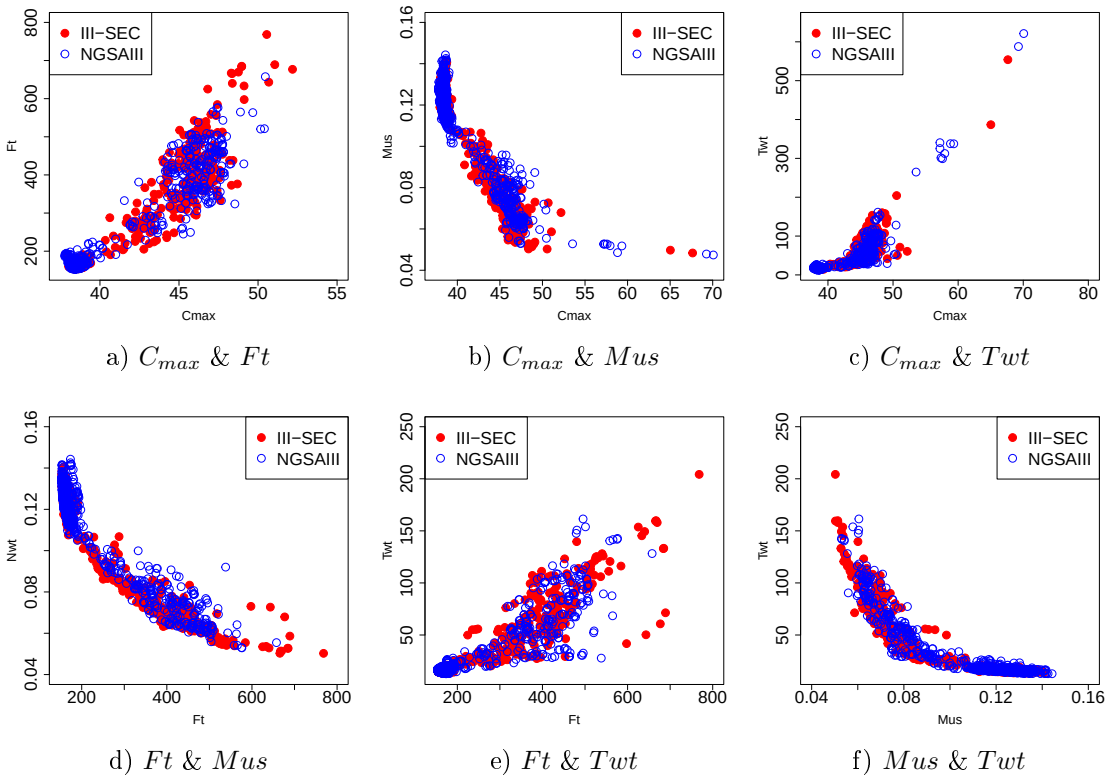
the other criteria, we see that ensembles provide better coverage of the objective space, which can be seen in Figures 8b, 8d, and 8f. Here it is interesting to observe that the DRs provide good coverage of the objective space for the standard scheduling criterion (C_max, Ft or Twt), which can be seen from the fact that many solutions are grouped in the space where these objectives are minimised. However, the remaining part of the objective space is poorly covered, and ensembles either achieve better solutions (as seen in Figure 8f) or better coverage of the objective space (seen in Figure 8d). This suggests that both DRs and ensembles perform equally well on combinations of criteria that do not conflict with each other, while ensembles can achieve solutions that offer a better trade-off between highly conflicting criteria.

6.4. Optimisation of five criteria

Table 5 shows the HV obtained when optimising the $R \mid r_j \mid C_{max}, Ft, Mus, Nwt, Twt$ problem. For the IND scenario, the ensembles can only match the performance of the individual DRs, while the ensembles in the ALL scenario perform significantly better in all situations, except when using the vote combination method with larger ensemble sizes. Both the ensembles created by SEC and the GAs achieve equally good results, with the ensembles created by SEC achieving a slightly better HV value.

Figure 9 shows the ability of ensembles to improve on the results achieved by individual rules, as the ensembles obtained a significantly better distribution of results for each configuration. In addition, the dispersion of results obtained by ensembles is lower than that of individual rules, which means that the algorithm is more stable and less likely to lead to scattered solutions. Smaller ensembles performed better than larger ones, meaning that there is no need to construct larger ensembles, which in turn has a positive impact on execution time as smaller ensembles can be constructed and interpreted more quickly. We also find here that the results gradually deteriorate as the size of the ensemble increases, especially in the case of the vote combination method. Finally, the ensembles created with the sum combination method perform better in all cases.

The Pareto fronts obtained by DRs and ensembles are outlined in Figure 10. For most pairs of cri-



a) C_{max} & Ft b) C_{max} & Mus c) C_{max} & Twt

d) Ft & Mus e) Ft & Twt f) Mus & Twt

Fig. 8. The Pareto front obtained for the $R \mid r_j \mid C_{max}, Ft, Mus, Twt$ problem denoted through pairwise combinations of the four optimised criteria Pareto front.

Table 5
Results for the HV metric for the $R \mid r_j \mid C_{max}, Ft, Mus, Nwt, Twt$ problem

Rules	RCM	ECM	sum			vote		
			3	5	7	3	5	7
IND	NSGA-II	E-NSGA-II	0.67 (≈|≈)	0.66 (≈|≈)	0.66 (≈|≈)	0.67 (≈|≈)	0.66 (≈|≈)	0.66 (≈|≈)
	NSGA-III	E-NSGA-III	0.66 (≈|≈)	0.66 (≈|≈)	0.65 (≈|≈)	0.66 (≈|≈)	0.66 (≈|≈)	0.65 (≈|≈)
	NSGA-II	SEC	0.68 (≈| +)	0.67 (≈|≈)	0.66 (≈|≈)	0.68 (≈| +)	0.67 (≈|≈)	0.67 (≈|≈)
	NSGA-III	SEC	0.66 (≈|≈)	0.66 (≈|≈)	0.67 (≈|≈)	0.67 (≈|≈)	0.67 (≈|≈)	0.66 (≈|≈)
ALL	NSGA-II	E-NSGA-II	0.70 (+ | +)	0.69 (+ | +)	0.68 (+ | +)	0.69 (+ | +)	0.68 (≈| +)	0.67 (≈|≈)
	NSGA-III	E-NSGA-III	0.68 (+ | +)	0.68 (+ | +)	0.68 (+ | +)	0.68 (+ | +)	0.67 (≈|≈)	0.66 (≈|≈)
	NSGA-II	SEC	0.70 (+ | +)	0.69 (+ | +)	0.69 (+ | +)	0.69 (+ | +)	0.68 (+ | +)	0.67 (≈| +)
	NSGA-III	SEC	0.69 (+ | +)	0.69 (+ | +)	0.69 (+ | +)	0.69 (+ | +)	0.68 (+ | +)	0.67 (≈|≈)
NSGA-II			0.66					
NSGA-III			0.65					

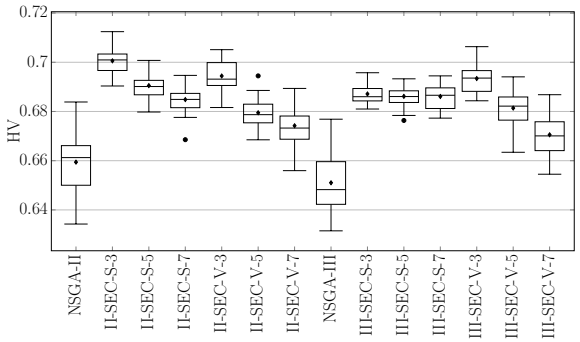


Fig. 9. Results for the HV metric for the $R \mid r_j \mid C_{max}, Ft, Mus, Nwt, Twt$ problem

teria, it can be seen that both rules and ensembles provide similar coverage of the objective space. However, some figures like Figure 10d, 10f, and 10i show that ensembles provide better coverage of the objective space, especially in the middle. We note that this is also the case when considering the Mus criterion, as it conflicts with the other scheduling criteria. Therefore, ensembles again provide better coverage of the objective space, especially for parts of the objective space that represent a compromise between conflicting criteria.

7. Further analysis and discussion

7.1. Ensemble composition analysis

In this section we analyse the composition of the ensembles created by SEC to see if certain observations can be made about their composition. Only the ALL scenario is considered, as better results were obtained in this case. Furthermore, the figures are only outlined for selected problems, as similar patterns were observed for all problems.

Figure 11 shows the frequency with which each DR is used in the ensembles constructed for the $R \mid r_j \mid C_{max}, Ft, Mus, Nwt, Twt$ problem, aggregated for ensemble combination methods and sizes. Although some DRs are used more frequently than others, the differences in their frequency of use are not large. On the contrary, most DRs are used to some extent to form ensembles, suggesting that no DR is inherently unsuitable to be used to form ensembles. Rather, it seems that any rule can be used to form high quality ensembles; this is probably why SEC performs better when using larger DR sets, as it can more easily find DRs that work well when combined in an ensemble.

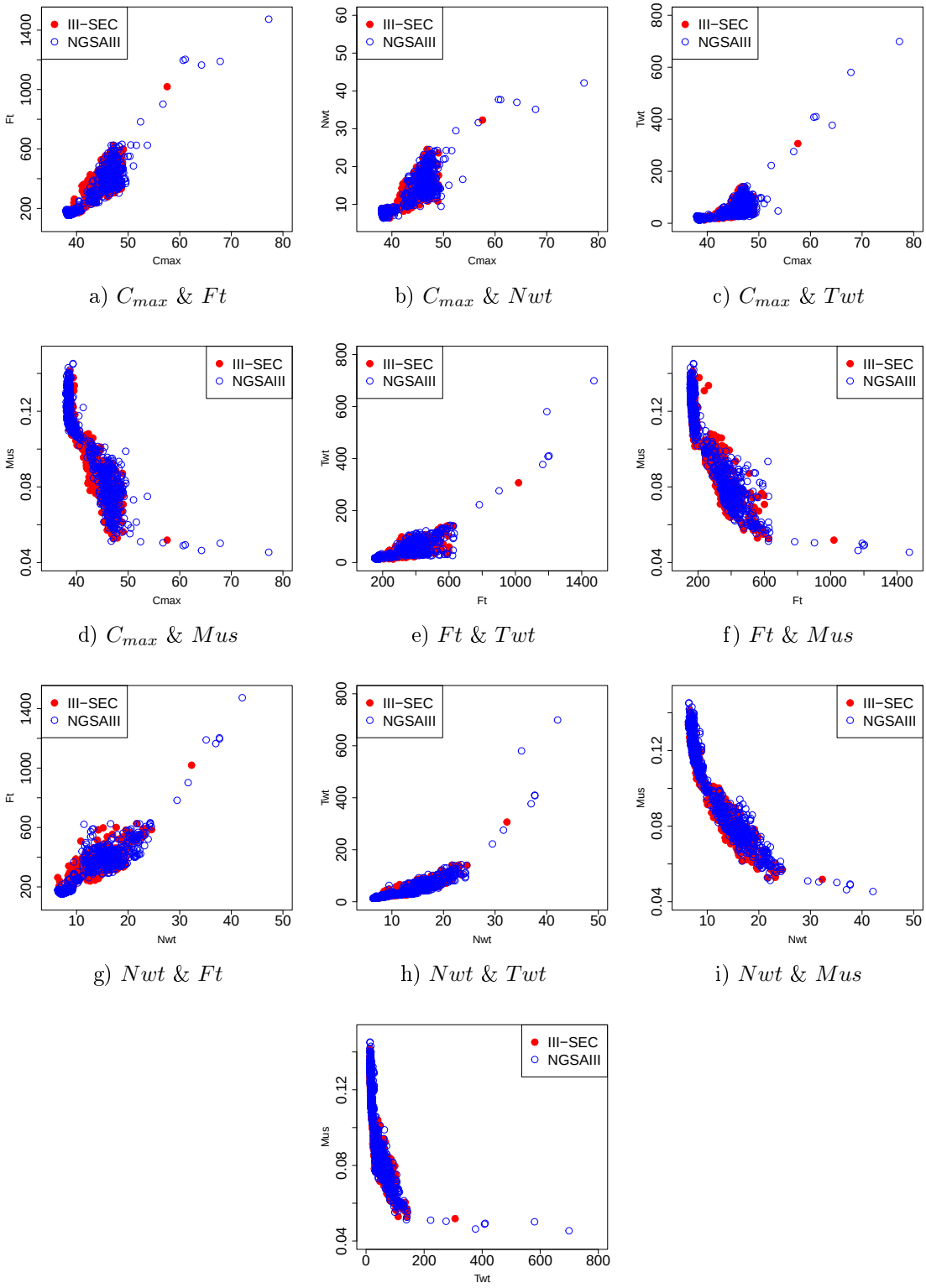
Table 6
Average Jaccard similarity values for ensembles.

Optimised criteria	Mean similarity
C_{max}, Twt	0.166
C_{max}, Mus, Twt	0.005
C_{max}, Ft, Mus, Twt	0.002
$C_{max}, Ft, Mus, Nwt, Twt$	0.001

Since more or less all DRs are used in the construction of ensembles, we ask whether there is some similarity between the constructed ensembles in the sense that they consist of similar DRs. For this purpose, we calculate the average Jaccard similarity index between all pairs of the constructed ensembles for all problems and present them in Table 6. The values range from 0 to 1, with 0 indicating no similarity and 1 indicating complete similarity between two groups. We note that only in the problem where two criteria are optimised is there a noticeable similarity, while in the larger problems there is almost no similarity between the ensembles. This is probably also due to the fact that for the two criteria problem there were only about 30 DRs available for ensemble construction, while for other problems this number was much higher (at least 700).

Another point of view from which the ensembles can be analysed is their relative performance compared to the DRs used to construct them. To this goal, we calculate the average performance of the DRs in the ensemble for all criteria and subtract it from the objective values achieved by the ensemble. A positive value means that the ensemble performs better, while a negative value means that the individual rules that make up the ensemble perform better on average. Table 7 shows the relative average values for all MO problems in %. From this we can see that the ensembles perform better on average than the DRs from which they were formed. We see that, with one exception, the ensembles perform significantly better on certain criteria than the individual rules that form them. In particular, for the Twt, Ft and Nwt criteria the ensembles perform better than the rules that make them up by about 50%, 20%, and 10%, respectively.

As an example, Table 8 shows the performance of a randomly selected ensemble and the



a) C_{max} & Ft b) C_{max} & Nwt c) C_{max} & Twt

d) C_{max} & Mus e) Ft & Twt f) Ft & Mus

g) Nwt & Ft h) Nwt & Twt i) Nwt & Mus

j) Twt & Mus

Fig. 10. The Pareto front obtained for the $R \mid r_j \mid C_{max}, Ft, Mus, Nwt, Twt$ problem denoted through pairwise combinations of the five optimised criteria Pareto front.

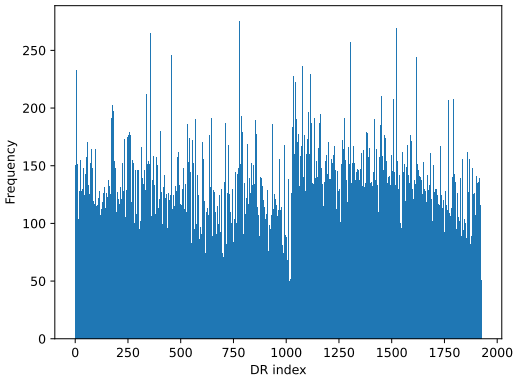


Fig. 11. Frequency of using DRs in ensembles

Table 7

Average relative difference between the objective values obtained by the ensemble and DRs it consists of

Optimised criteria	Twt	Nwt	Ft	C_{max}	Mus
C_{max}, Twt	5.80	-	-	0.01	-
C_{max}, Mus, Twt	57.87	-	-	2.15	4.98
C_{max}, Ft, Mus, Twt	58.55	-	22.17	1.79	0.27
$C_{max}, Ft, Mus, Nwt, Twt$	51.76	13.24	22.51	2.11	-0.26

5 DRs that make it up for the $R \mid r_j \mid C_{max}, Ft, Mus, Nwt, Twt$ problem, with the best value for each criterion outlined in bold. The table shows that the ensemble performs best for 3 of the 5 criteria considered. However, what is more interesting is the observation that even though some rules perform poorly on certain criteria, the ensemble performs well on all five criteria considered. The worst performance is obtained for the Mus criterion, which is to be expected since further improvement on this criterion would probably lead to a deterioration of the results on all other criteria. However, this also shows that good ensembles can be formed from DRs that do not necessarily perform well on all criteria, which means that the fitness of individual DRs is not a reliable indicator of how the ensemble will perform.

7.2. Runtime analysis

This section outlines the runtime of the proposed method. The runtime can be divided into two independent parts: the time required to construct the ensembles and the execution time of the ensemble in solving a problem. The first part is more computationally intensive, as a large number

Table 8

Performance of a selected ensemble and the individual rules from which it is constructed.

		Twt	Nwt	Ft	C_{max}	Mus
Ensemble		**17.15**	8.23	**161.81**	**39.24**	0.13
DR index	424	18.33	8.41	164.39	39.59	0.12
	1260	70.50	16.94	442.89	49.17	0.07
	658	20.29	9.21	198.74	40.67	0.10
	1503	102.90	21.11	486.93	48.16	**0.06**
	589	19.24	**7.78**	168.47	40.76	0.14

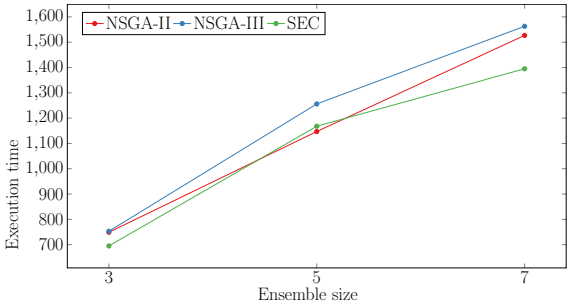


Fig. 12. Execution times of algorithms for ensemble construction

of ensembles must be constructed to obtain a good quality Pareto front. Figure 12 outlines the execution time of SEC and the two GAs when constructing ensembles. It shows that already for the lowest ensemble size more than 10 minutes are needed to construct the ensembles (given that 10 000 ensembles are constructed with the methods) and that the execution time increases linearly with the increase of the ensemble size. However, this part can be executed offline before a specific problem has to be solved, and ensembles can be prepared in advance.

Then, when a new scheduling problem instance needs to be solved, a suitable ensemble is selected from the set of prepared ones and applied on the problem to create the schedule. In this case, the execution time for the ensemble is almost negligible, as a single ensemble takes about 0.1-0.2 seconds to solve all 60 instances in the test set. This means that the time required for the ensemble to make a single decision is even lower, meaning it can be used to solve dynamic problems in real time.

7.3. Result discussion

When comparing the results of SEC and GAs for creating ensembles, no significant difference was found, which means that both methods work equally well. However, SEC offers a few slight advantages over GAs. Firstly, SEC is simpler as this method randomly samples ensembles and non-dominantly sorts the set of solutions obtained. The GAs contain additional elements, such as mutation and crossover operators, that need to be defined and a method for determining the diversity of solutions in the objective space. With SEC, only the ensemble-related parameters (combination method and size) and the stopping condition need to be defined, whereas with GAs, additional parameters such as population size, mutation probability, tournament size and the like need to be selected. Finally, SEC is also somewhat less computationally intensive due to its inherent simplicity, as can be seen from the runtime analysis. This demonstrates that SEC is equally good in constructing ensembles as other more complex optimisation methods.

Although the results obtained in the experimental analysis demonstrate that ensembles improve the performance of individual rules, there are several factors that influence their performance. First, a large set of diverse DRs is required to construct high quality ensembles, otherwise no improvements are obtained as demonstrated in the IND scenario where a Pareto front from a single run was used for ensemble construction. It seems that an individual Pareto front does not offer the necessary diversity between DRs, thus rules obtained from Pareto fronts of several executions should be used together to obtain better results.

Regarding the parameters for creating ensembles, the best results were obtained when smaller ensemble sizes were used, most frequently with size 3. Moreover, the results generally deteriorated as the size of the ensemble increased, which was not observed with the SO optimisation [18]. The reason why the results deteriorate with increasing ensemble size is that it becomes more difficult to find a combination of ensembles that work well together. This suggests that it is more difficult for the ensemble construction methods to find a larger number of rules that work well together, especially since there is a large set of rules to choose from and the selection of rules is random. A possible remedy would be to use a more sophisticated method to select the rules to form the ensemble. However, the ensemble composition analysis conducted earlier did not reveal any regularities in the structure of ensembles that could be used for this purpose, so this needs to be investigated further. Regardless, the strategy of random selection for the composition of the ensemble has already proven sufficient to achieve significant performance improvements over individual DRs.

As far as the ensemble combination method is concerned, neither method consistently performs better. However, the results show that the performance of the vote combination method deteriorates more as the size of the ensemble increases. While similar behaviour is observed with the sum method, the deterioration in results is smaller and larger ensembles generally perform better with the sum method than with the vote combination method. The reason why the vote method is more prone to this behaviour is because of the way it makes the decisions. With the vote combination method, it is possible that adding a rule to the ensemble does not change its decisions if it is outvoted by all other rules in the ensemble. Although a similar situation can occur with the sum combination method, it is not as pronounced. Thus, it is more difficult for the vote method to construct ensembles that provide the required trade-off between the considered objective.

Finally, the visualisations of the Pareto fronts between ensembles and individual rules show that for most problems ensembles achieve Pareto fronts with better coverage of the objective space. Although ensembles and rules obtain equally good solutions at the extremes of each criterion, ensembles obtain solutions that offer a better trade-off between different criteria. This is particularly evident when optimising conflicting criteria. In this case, ensembles provide a better coverage of the objective space, which is a significant advantage in multi-objective optimisation.

8. Conclusion

This study investigated the use of ensembles of DRs in the context of multi-objective scheduling problems, motivated by the fact that there is a limit to the performance of a single DR. Creating ensembles of DRs has already been shown to

be a fairly efficient method for improving the performance of DRs. However, so far they have not been used to improve the performance of MO DRs. Therefore, in this study, two different methods for creating MO ensembles of DRs were proposed and evaluated on several MO scheduling problems with different numbers of criteria to be optimised.

The results show that ensembles improve the performance of individual rules for MO problems, as evidenced by achieving better coverage of the objective space. Although the method performs well on all problems tested, the main advantage of ensembles over individual rules becomes more evident as the number of optimised criteria increases. The results also show that the SEC method performs as well as more sophisticated GAs, proving to be well suited for ensemble construction even for MO problems. However, the performance of ensemble construction methods is largely influenced by several elements. First of all, a large and diverse set of DRs must be used for ensemble construction, otherwise no improvements can be achieved. In addition, the best results were obtained with smaller ensembles, highlighting the difficulty of finding larger rule sets that work well together. Although a possible remedy would be to use a more sophisticated DR selection strategy to construct the ensembles, the analysis of the structure of the ensembles did not reveal any regularities that could be used for this purpose. Therefore, further investigation is needed to resolve the problems observed in some situations.

In future studies, we plan to take the research in several directions. One goal would be to investigate SEC with other ensemble construction methods specific to MO problems, which would guide the ensemble construction method to select those rules that better cover certain parts of the search space. This would serve to reduce the randomness in the selection process of the DRs that make up the ensemble Another direction would be to apply the method to other problems, such as the vehicle routing problem, to test its performance on different combinatorial problems. It would also be interesting to investigate alternative ensemble combination methods, apart from the most commonly used sum and vote methods, such as Borda counting and the like Finally, we also plan to develop a hybrid algorithm that can simultaneously evolve both ensembles and DRs. In this case, the algorithm can focus more on developing DRs that are more suitable for use in ensembles, rather than using rules developed for individual use.

Acknowledgements

This work has been supported in part by Croatian Science Foundation under the project IP-2019-04-4333 and by the Spanish Government under research project PID2019-106263RB-I00.

References

[1] Pinedo ML. Scheduling. Springer US; 2012.

[2] Hart E, Ross P, Corne D. Evolutionary Scheduling: A Review. Genetic Programming and Evolvable Machines. 2005 Jun;6(2):191-220.

[3] Đurasević M, Jakobović D. A survey of dispatching rules for the dynamic unrelated machines environment. Expert Systems with Applications. 2018;113:555-69.

[4] Poli R, Langdon WB, McPhee NF. A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd; 2008.

[5] Gil-Gala FJ, Varela R. Genetic Algorithm to Evolve Ensembles of Rules for On-Line Scheduling on Single Machine with Variable Capacity. In: From Bioinspired Systems and Biomedical Applications to Machine Learning. Springer International Publishing; 2019. p. 223-33.

[6] Nguyen S, Zhang M, Johnston M, Tan KC. Dynamic Multi-objective Job Shop Scheduling: A Genetic Programming Approach. In: Studies in Computational Intelligence. Springer Berlin Heidelberg; 2013. p. 251-82.

[7] Đurasević M, Jakobović D, Knežević K. Adaptive scheduling on unrelated machines with genetic programming. Applied Soft Computing. 2016;48:419-30.

[8] Đumić M, Šišejković D, Čorić R, Jakobović D. Evolving priority rules for resource constrained project scheduling problem with genetic programming. Future Generation Computer Systems. 2018;86:211-21.

[9] Branke J, Nguyen S, Pickardt CW, Zhang M. Automated Design of Production Scheduling Heuristics: A Review. IEEE Transactions on Evolutionary Computation. 2016;20(1):110-24.

[10] Nguyen S, Zhang M, Johnston M, Tan KC. Automatic Design of Scheduling Policies for Dynamic Multi-objective Job Shop Scheduling via Cooperative Coevolution Genetic Programming. IEEE Transactions on Evolutionary Computation. 2014 Apr;18(2):193-208.

[11] Đurasević M, Jakobović D. Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey. Artificial Intelligence Review. 2022 Aug.

[12] Masood A, Mei Y, Chen G, Zhang M. Many-objective genetic programming for job-shop scheduling; 2016. p. 209-16.

[13] Đurasević M, Jakobović D. Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. Genetic Programming and Evolvable Machines. 2017 Sep;19(1-2):9-51.

[14] Zhang F, Mei Y, Nguyen S, Tan KC, Zhang M. Multitask Genetic Programming-Based Generative Hyperheuristics: A Case Study in Dynamic Scheduling. IEEE Transactions on Cybernetics. 2021:1-14.

[15] Zhang F, Mei Y, Nguyen S, Zhang M, Tan KC. Surrogate-Assisted Evolutionary Multitask Genetic Programming for Dynamic Flexible Job Shop Scheduling. IEEE Transactions on Evolutionary Computation. 2021;25(4):651-65.

[16] Đurasević M, Jakobović D. Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics. Journal of Computational Science. 2022;61:101649.

[17] Park J, Nguyen S, Zhang M, Johnston M. Evolving Ensembles of Dispatching Rules Using Genetic Programming for Job Shop Scheduling; 2015. p. 92-104.

[18] Đurasević M, Jakobović D. Creating dispatching rules by simple ensemble combination. Journal of Heuristics. 2019 May;25(6):959-1013.

[19] Đurasević M, Planinić L, Gil-Gala FJ, Jakobović D. Constructing Ensembles of Dispatching Rules for Multi-objective Problems. In: Ferrández Vicente JM, Álvarez-Sánchez JR, de la Paz López F, Adeli H, editors. Bio-inspired Systems and Applications: from Robotics to Ambient Intelligence. Cham: Springer International Publishing; 2022. p. 119-29.

[20] Burke EK, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E, Qu R. Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society. 2013;64(12):1695-724.

[21] Nguyen S, Mei Y, Zhang M. Genetic programming for production scheduling: a survey with a unified framework. Complex & Intelligent Systems. 2017 Feb;3(1):41-66.

[22] Liu Y, Mei Y, Zhang M, Zhang Z. Automated Heuristic Design Using Genetic Programming Hyper-Heuristic for Uncertain Capacitated Arc Routing Problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '17. New York, NY, USA: Association for Computing Machinery; 2017. p. 290–297.

[23] Wang S, Mei Y, Zhang M, Yao X. Genetic Programming With Niching for Uncertain Capacitated Arc Routing Problem. IEEE Transactions on Evolutionary Computation. 2022;26(1):73-87.

[24] Zhang F, Mei Y, Nguyen S, Zhang M. Collaborative Multifidelity-Based Surrogate Models for Genetic Programming in Dynamic Flexible Job Shop Scheduling. IEEE Transactions on Cybernetics. 2021:1-15.

[25] Zhang F, Mei Y, Nguyen S, Zhang M. Evolving Scheduling Heuristics via Genetic Programming With Feature Selection in Dynamic Flexible Job-Shop Scheduling. IEEE Transactions on Cybernetics. 2021;51(4):1797-811.

[26] Zhang F, Mei Y, Zhang M. A Two-Stage Genetic Programming Hyper-Heuristic Approach with Feature Selection for Dynamic Flexible Job Shop Scheduling. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '19. New York, NY, USA: Association for Computing Machinery; 2019. p. 347–355.

[27] Zhang F, Nguyen S, Mei Y, Zhang M. Multitask Learning in Hyper-Heuristic Domain with Dynamic Production Scheduling. In: Genetic Programming for Production Scheduling. Springer Singapore; 2021. p. 249-69.

[28] Jaklinović K, Đurasević M, Jakobović D. Designing dispatching rules with genetic programming for the unrelated machines environment with constraints. Expert Systems with Applications. 2021;172:114548.

[29] Judt D, Lawson C, van Heerden AS. Rapid design of aircraft fuel quantity indication systems via multi-objective evolutionary algorithms. Integrated Computer-Aided Engineering. 2021;28(2):141-58.

[30] Bai Q, Miralinaghi M, Labi S, Sinha KC. Methodology for analyzing the trade-offs associated with multi-objective optimization in transportation asset management under uncertainty. Computer-Aided Civil and Infrastructure Engineering. 2021;36(4):381-401.

[31] Xue Y, Jiang P, Neri F, Liang J. A multi-objective evolutionary approach based on graph-in-graph for neural architecture search of convolutional neural networks. International Journal of Neural Systems. 2021;31(09):2150035.

[32] Liang Y, He F, Zeng X, Luo J. An improved loop subdivision to coordinate the smoothness and the number of faces via multi-objective optimization. Integrated Computer-Aided Engineering. 2022;(Preprint):1-19.

[33] Gutierrez Soto M, Adeli H. Many-objective control optimization of high-rise building structures using replicator dynamics and neural dynamics model. Structural and Multidisciplinary Optimization. 2017;56(6):1521-37.

[34] Civera M, Pecorelli ML, Ceravolo R, Surace C, Zanotti Fragonara L. A multi-objective genetic algorithm strategy for robust optimal sensor placement. Computer-Aided Civil and Infrastructure Engineering. 2021;36(9):1185-202.

[35] Rodrigues D, Papa JP, Adeli H. Meta-heuristic multi- and many-objective optimization techniques for solution of machine learning problems. Expert Systems. 2017;34(6):e12255.

[36] Xue Y, Zhu H, Neri F. A self-adaptive multi-objective feature selection approach for classification problems. Integrated Computer-Aided Engineering. 2022;(Preprint):1-19.

[37] Tay JC, Ho NB. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Computers & Industrial Engineering. 2008;54(3):453-73.

[38] Zhang F, Mei Y, Zhang M. Evolving Dispatching Rules for Multi-objective Dynamic Flexible Job Shop Scheduling via Genetic Programming Hyper-heuristics. In: 2019 IEEE Congress on Evolutionary Computation (CEC); 2019. p. 1366-73.

[39] Zhang F, Nguyen S, Mei Y, Zhang M. Learning Scheduling Heuristics for Multi-objective Dynamic Flexible Job Shop Scheduling. In: Genetic Program-

ming for Production Scheduling. Springer Singapore; 2021. p. 235-45.

[40] Masood A, Chen G, Mei Y, Al-Sahaf H, Zhang M. Genetic Programming with Pareto Local Search for Many-Objective Job Shop Scheduling. In: AI 2019: Advances in Artificial Intelligence. Springer International Publishing; 2019. p. 536-48.

[41] Masood A, Chen G, Mei Y, Al-Sahaf H, Zhang M. A Fitness-based Selection Method for Pareto Local Search for Many-Objective Job Shop Scheduling. In: 2020 IEEE Congress on Evolutionary Computation (CEC); 2020. p. 1-8.

[42] Xu B, Mei Y, Wang Y, Ji Z, Zhang M. Genetic Programming with Delayed Routing for Multiobjective Dynamic Flexible Job Shop Scheduling. Evolutionary Computation. 2021 03;29(1):75-105.

[43] Sagi O, Rokach L. Ensemble learning: A survey. WIREs Data Mining and Knowledge Discovery. 2018 Feb;8(4).

[44] Nandi A, Xhafa F, Subirats L, Fort S. Reward-Penalty Weighted Ensemble for Emotion State Classification from Multi-Modal Data Streams. International journal of neural systems. 2022:2250049-9.

[45] Avola D, Cascio M, Cinque L, Fagioli A, Foresti GL. LieToMe: An ensemble approach for deception detection from facial cues. International Journal of Neural Systems. 2021;31(02):2050068.

[46] Ganaie MA, Hu M, Malik AK, Tanveer M, Suganthan PN. Ensemble deep learning: A review. Engineering Applications of Artificial Intelligence. 2022 Oct;115:105151.

[47] Alam KM, Siddique N, Adeli H. A dynamic ensemble learning algorithm for neural networks. Neural Computing and Applications. 2020;32(12):8675-90.

[48] Benamara NK, Val-Calvo M, Alvarez-Sanchez JR, Diaz-Morcillo A, Ferrandez-Vicente JM, Fernandez-Jover E, Stambouli, TB. Real-time facial expression recognition using smoothed deep neural network ensemble. Integrated Computer-Aided Engineering. 2021;28(1):97-111.

[49] Gasienica-Józkowy J, Knapik M, Cyganek B. An ensemble deep learning method with optimized weights for drone-based water rescue and surveillance. Integrated Computer-Aided Engineering. 2021;28(3):221-35.

[50] Park J, Mei Y, Nguyen S, Chen G, Johnston M, Zhang M. Genetic Programming Based Hyper-heuristics for Dynamic Job Shop Scheduling: Cooperative Coevolutionary Approaches. In: Lecture Notes in Computer Science. Springer International Publishing; 2016. p. 115-32.

[51] Hart E, Sim K. A Hyper-Heuristic Ensemble Method for Static Job-Shop Scheduling. Evolutionary Computation. 2016;24(4):609-35.

[52] Park J, Mei Y, Nguyen S, Chen G, Zhang M. An Investigation of Ensemble Combination Schemes for Genetic Programming based Hyper-heuristic Approaches to Dynamic Job Shop Scheduling. Applied Soft Computing. 2017 11;63.

[53] Đurasević M, Jakobović D. Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. Genetic Programming and Evolvable Machines. 2017 Apr;19(1-2):53-92.

[54] Đumić M, Jakobović D. Ensembles of priority rules for resource constrained project scheduling problem. Applied Soft Computing. 2021;110:107606.

[55] Gil-Gala FJ, Sierra MR, Mencía C, Varela R. Combining hyper-heuristics to evolve ensembles of priority rules for on-line scheduling. Natural Computing. 2020 Jun.

[56] Gil-Gala FJ, Mencía C, Sierra MR, Varela R. Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms. Integrated Computer-Aided Engineering. 2021;28(1):65-80.

[57] Đurasević M, Planinić L, Gala FJG, Jakobović D. Novel Ensemble Collaboration Method for Dynamic Scheduling Problems. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '22. New York, NY, USA: Association for Computing Machinery; 2022. p. 893–901.

[58] Đurasević M, Jakobović D. Comparison of schedule generation schemes for designing dispatching rules with genetic programming in the unrelated machines environment. Applied Soft Computing. 2020;96:106637.

[59] Planinić L, Backović H, Đurasević M, Jakobović D. A Comparative Study of Dispatching Rule Representations in Evolutionary Algorithms for the Dynamic Unrelated Machines Environment. IEEE Access. 2022;10:22886-901.

[60] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation. 2002;6(2):182-97.

[61] Deb K, Jain H. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. IEEE Transactions on Evolutionary Computation. 2014;18(4):577-601.

[62] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation. 1999;3(4):257-71.

[63] Riquelme N, Lucken CV, Baran B. Performance metrics in multi-objective optimization. In: 2015 Latin American Computing Conference. IEEE; 2015. .

[64] Audet C, Bigeon J, Cartier D, Digabel SL, Salomon L. Performance indicators in multiobjective optimization. European Journal of Operational Research. 2021 Jul;292(2):397-422.