Combining Single Objective Dispatching Rules into Multi-objective Ensembles for the Dynamic Unrelated Machines Environment

Marko Đurasević[a,*], Francisco Javier Gil Gala[b], Domagoj Jakobović[a], Carlos A. Coello Coello[c]

[a] *University of Zagreb Faculty of Electrical Engineering and Computing, Zagreb, Croatia*
[b] *Department of Computer Science, University of Oviedo, Campus de Viesques s/n, Gijón, 33271, Spain*
[c] *Department of Computer Science, CINVESTAV-IPN (Evolutionary Computation Group), Mexico City, Mexico*

Abstract

Dispatching rules (DRs), which are simple constructive methods that incrementally build the schedule, represent the most popular method for solving dynamic scheduling problems. These DRs were usually designed for optimising a single criterion and work poorly when solving multi-objective (MO) problems. Therefore, in recent years, we have seen an increase of research dealing with automated design of DRs using genetic programming (GP), which has enabled the application of several evolutionary MO optimisation methods to create DRs for MO problems. However, for each considered MO problem new DRs need to be evolved, which can be computationally expensive. Motivated by this, we propose a novel methodology to combine existing DRs evolved for optimising individual criteria into ensembles appropriate for optimising multiple criteria simultaneously. For this purpose, we adapt the existing simple ensemble construction (SEC) method to construct ensembles of DRs for optimising MO problems. The method is evaluated on several MO scheduling problems and

[⋆] Fully documented templates are available in the elsarticle package on CTAN.

[*] Corresponding author

Email addresses: support@elsevier.com (Francisco Javier Gil Gala), support@elsevier.com (Domagoj Jakobović), support@elsevier.com (Carlos A. Coello Coello)

URL: www.elsevier.com (Marko Đurasević)

compared with DRs evolved by NSGA-II and NSGA-III. The obtained results show that for most problems the proposed method constructed ensembles that significantly outperform DRs developed with standard MO algorithms. Furthermore, we propose the application of evolved MO rules on problems with a smaller number of criteria and demonstrate that with such a strategy similar or better performance is achieved compared to evolving DRs for such problems directly, which demonstrates the reusability and generalisation potential of the evolved DRs.

Keywords: Dispatching rules, hyper-heuristic, multi-objective optimisation, ensembles, unrelated machines environment

---

1. Introduction

Scheduling is an important decision making process in which a set of jobs is allocated to a limited number of machines for execution [1]. Scheduling problems appear in many real-world situations, like multiprocessor scheduling [2], equipment scheduling [3], manufacturing [4], or power scheduling [5, 6]. Since most scheduling problems of interest are NP-hard, they have been often solved using various metaheuristic methods [7], such as genetic algorithms [8], ant colony optimisation [9, 10], simulated annealing [11, 12], differential evolution [13], and a variety of local search based methods [14, 15, 16], among many others. However, metaheuristics can only be applied for solving static scheduling problems, in which all the information about the problem is known beforehand. Since many real-world problems are dynamic, meaning that not all information about the problem is known beforehand, metaheuristic methods cannot be usually applied for solving them. As a consequence, many simple constructive heuristics have been proposed for solving such problems [17].

In the context of scheduling problems, these simple constructive heuristics are called dispatching rules (DRs) [1]. In order to be applicable for dynamic environments, DRs perform only the next scheduling decision, and usually do not schedule jobs in advance, but only when a certain machine becomes free.

However, certain limitations and issues are associated with DRs. Efficient DRs are difficult to design, even when considering simple single objective problems. Furthermore, due to their myopic view on the problem, their performance is limited.

In order to address the previous issues, the automated development of DRs has garnered the attention of numerous researchers [18, 19]. In that regard, a lot of studies focused on applying various hyper-heuristic methods to automatically design DRs for different scheduling problems. Out of the numerous methods that were applied for automated design of DRs, genetic programming (GP) [20] profiled itself as the dominant hyper-heuristic method for that purpose. Using GP, DRs can be created faster without a long trial and error process, but also for various scheduling problem variants [21, 22, 23, 24, 25].

Currently, one important research area is multi-objective (MO) optimisation, since MO problems appear in many areas, such as energy saving [26], industrial design [27], scheduling [28], nuclear engineering [29] and engineering in general [30]. Although over the years many methods were proposed for solving MO problems [31, 32], many issues and research directions still remain, such as improving performance, reducing computational complexity, handling premature convergence, and others [33]. As a result, MO optimisation is also a highly investigated topic in the unrelated machines environment [34]. Currently, there is a lack of manually designed DRs for such problems [34], which is expected as they are difficult to design. This can be resolved using GP coupled with various MO algorithms like NSGA-II or NSGA-III to evolve DRs, which in several occasions demonstrated to achieve very good performance [22, 35]. However, this can be time consuming since the evolutionary process needs to be executed for each new criteria combination. This motivates us to raise the question of whether it would be possible to avoid the evolutionary process in its entirety, assuming that rules for optimising single-objective (SO) problems already exist. One such possibility would be to combine existing rules to perform scheduling decisions collectively, but in a way that they do not optimise only a single criteria, but rather several criteria simultaneously. This idea is inspired by the application of ensemble learning for various SO scheduling problems, for which it was already shown that DRs can be used collectively to further improve their performance [36, 37].

In this paper, we investigate the possibility of creating ensembles specialised for solving MO optimisation scheduling problems using DRs that were evolved only for optimising a single criterion. For that purpose, we apply existing methods for constructing ensembles out of existing DRs like the SEC method [37], and also a new GA adapted for creating ensembles of DRs. The proposed methodology is then applied for constructing ensembles of DRs for MO optimisation in the context of the unrelated machines environment. The method is compared to MO DRs evolved by standard MO algorithms like NSGA-II and NSGA-III, to evaluate its performance. We further perform several additional analyses on the results to gain a better understanding of the proposed methodology, and the possibility of reusing rules and ensembles evolved for one MO problem on other larger or smaller MO problems. Thus, the core contributions of this paper can be summarised as follows:

1. A novel methodology of applying DRs evolved for a single objective for MO optimisation by combining them into ensembles.
2. Adaptation of the Simple Ensemble Combination (SEC), NSGA-II and NSGA-III methods to construct ensembles of DRs for MO problems.
3. Strategy to reuse evolved Pareto sets of ensembles and rules for optimising other MO problems with a larger or smaller number of criteria.

The rest of the paper is organised as follows. The literature review is given in Section 2. The unrelated machines scheduling environment is described in Section 3. The proposed procedure of constructing MO ensembles out of DRs evolved for individual criteria is described in Section 4. Section 5 describes the experimental setup, whereas Section 6 outlines the obtained results. A further analysis of the proposed method and the obtained results is provided in Section 7, whereas Section 8 gives a short summary of the main findings and provides a brief discussion on them. Finally, the conclusion and directions for future work are outlined in Section 9.

2. Literature Review

In order to design novel heuristics, numerous hyper-heuristic methods have been applied over the years [38]. One of the most popular hyper-heuristic methods is genetic programming (GP), which has been applied to generate heuristics for a wide range of problems, including scheduling [18], vehicle routing problems [39], travelling salesman problem [40], and capacitated arc routing [41]. This is especially true in the context of scheduling problems, where GP has been widely used to automatically generate new DRs for various problem variants that include the single machine problem [24], unrelated machines scheduling problem [21], job shop [42], and resource constrained scheduling problem [25, 43]. Recent years saw many new research directions being explored in the area of automated design of DRs, some of which include: application of surrogate models [44], feature selection [45], local search [46], multitask GP [47], and many others [48, 49].

As this study is focused on tackling MO problems using ensemble learning methods in the context of hyper-heuristics, the rest of the section provides a detailed review of literature dealing with the automated design of MO rules and creating ensembles of DRs.

2.1. Automated design of MO DRs

The first study dealing with MO optimisation in the context of automated DR design was done by Nguyen et al. [22], in which the authors optimised five objectives simultaneously. The authors applied the HaD-MOEA algorithm to obtain Pareto fronts of DRs for the considered problem, and compared the obtained rules to several manually designed DRs to illustrate their performance. This initial study showed that MO algorithms can effectively be coupled with GP to automatically design DRs for optimising multiple objectives simultaneously. In [50], the authors extended the previous study by employing a local search operator during the evolutionary process. This small addition to the algorithm led to improved quality of the evolved MO DRs. The previous studies were extended in [51] by considering two problems in which 4 and 5 criteria need to be optimised simultaneously. The authors employed several popular MO algorithms, such as SPEA2, NSGA-II, and NSGA-III to evolve MO DRs, and the results show that the best Pareto fronts were obtained using NSGA-III.

Several MO problems in which between 3 and 9 criteria were optimised using 4 MO algorithms were considered in [35]. The main focus of this study was to investigate different MO problems and examine whether for these problems it is possible to evolve DRs that perform better than existing (manually designed) DRs. The results indicated that the evolved rules were again better than the manually designed ones, especially for problems with fewer criteria. However, as the number of criteria that were optimised simultaneously increased, it also became more difficult for MO algorithms to obtain rules that performed better than manually designed rules across all criteria. In [52] and [53] the authors considered MO problems consisting of several flowtime related criteria and showed the effectiveness of NSGA-II for solving them. A Pareto local search method was coupled to GP to automatically design new DRs in [54] and [55]. The authors minimised four objectives simultaneously, and showed that the method based on Pareto local search performs better than GP coupled with standard MO methods like NSGA-III.

2.2. Ensembles of DRs

The methods for creating ensembles of DRs can be categorised as methods that generate the rules that will be contained in the ensembles, and those which use an existing set of pre-evolved rules to construct ensembles. Both types of methods have been investigated in the literature, and have been found to achieve better performance than individual DRs.

The first study dealing with the design of ensembles was that of Park et al. [36], in which the authors proposed a cooperative coevolutionary method to design ensembles of DRs. In the proposed method the population was divided into several sub-populations, and each sub-population was tasked with the evolution of a single rule contained in the ensemble. As such, this method immediately evolved the rules that would be contained in the ensemble. Each rule was evolved independently and only interacted with rules from other subpopulations when it was required to evaluate it. During the evaluation, the rules were combined into ensembles that used a simple vote mechanism between the rules to determine the next scheduling decision. The experiments showed that the evolved ensembles achieved better results than individual DRs. This research was further extended in [56] by applying a multilevel GP. In this method, the evolutionary process at each generation is divided into two phases: the first in which groups of rules are evolved, and the second in which the individual rules are evolved. The results show that the multilevel GP method does not outperform the cooperative coevolution method from [36]. However, it can construct ensembles significantly faster.

A novel ensemble learning method called NELLI-GP was applied in [57] for the evolution of ensembles. The goal of this method is to evolve individual DRs that are specialised for solving a subset of problem instances, and are then combined into ensembles. The rules contained in the ensemble are applied individually in a cyclic manner to construct the solution of a problem instance by applying each rule to perform a single scheduling decision. With this strategy the authors outperformed previous results from [36]. The authors also investigated various aspects of their method, demonstrating that the best results were obtained if larger ensembles (consisting of more than 60 rules) of DRs are constructed. This can naturally lead to ensembles and rules that are more difficult to interpret and slower to execute than smaller ones.

The way in which the rules in the ensemble are combined to reach a common decision represents an important design decision. Therefore, in [58] the authors performed an in-depth analysis of four methods for aggregating the individual decisions of rules into a single one: sum, vote, weighted sum, and weighted vote. The authors used the ensemble generation method proposed in [36] and combined it with the four ensemble combination methods. Their experimental results showed that the weighted combination methods (weighted sum and vote) perform worse than the standard combination methods (sum and vote). Furthermore, the results also suggested that the sum method achieves a better performance than the vote method. As such, this study outlines the importance of selecting the appropriate way of combining decisions of individual rules into a single decision.

In [59] the authors compared four ensemble construction methods, which include SEC, BagGP, BoostGP, and cooperative coevolution. The cooperative coevolution algorithm is similar to the method proposed in [36], whereas the BagGP and BoostGP methods are inspired by equivalent methods from machine learning, which either use sampling with repetition when constructing the training set (BagGP), or introduce weights for individual problem instances based on how well or how poorly they are solved by the ensemble (BoostGP). All three previous methods generate new DRs when constructing the ensembles. On the other hand, SEC uses a set of existing pre-evolved DRs to construct ensembles. The results indicate that the proposed SEC method achieved the best results among all the tested methods, which suggests that it might be reasonable to divide the evolution of DRs and the construction of ensembles into two independent processes. In [60] the authors test the previous methods on the resource constrained project scheduling problem and achieve similar results, which further confirms the conclusions of the previous study. As a result, the SEC method and its resilience were further investigated in [37], where several new heuristic ensemble construction methods were proposed and it was shown that the methods performed well even when using DRs evolved with different GP variants and parameters.

A different type of ensemble was investigated in [24], in which the rules contained in the ensemble were used independently from each other, meaning that their decisions were not aggregated into a single one. Instead, each rule was used to individually solve the considered problem instance, and then the best solution was selected. As such, the problem can actually be formulated as finding the minimal set of rules which will perform well on as many problem instances as possible. The reason why it was possible to use this strategy is because the problem under consideration was static. However, due to certain intrinsic properties of the problem it still had to be solved in a short amount of time. This type of ensemble was further studied in [61], where they were compared to ensembles constructed out of the manually designed ATC rule. The results of this study show that constructing ensembles from existing manually designed rules is not efficient, and that automatically designed DRs are better suited for that purpose. Furthermore, in [62] the authors applied different methods to construct such ensembles. The methods ranged from a simple greedy method, to a more sophisticated memetic algorithm, which achieved the best overall results. To remedy the fact that the previous kind of ensembles are not usable for dynamic problems, in [63] the authors adapted this type of ensemble in such a way that they construct a partial schedule considering only jobs that are released until the current point in time. With that adaptation, the ensembles can be used for dynamic problems. However, they achieved a better performance than ensembles using the standard vote and sum combination methods.

To the best of the authors' knowledge, the only study which has investigated the application of ensembles for MO problems is [64]. In this study, SEC was applied to construct ensembles out of existing MO DRs evolved by different MO algorithms, to further improve their performance. As such, this study already shows that ensembles can be applied in the context of MO optimisation, although the study considered only a single problem, and no detailed investigation was performed. However, the main difference between [64] and this study lies in the fact that in this study we do not wish to evolve new MO DRs, but rather reutilise existing SO rules for MO optimisation by combining them into ensembles. Thus, in this study we propose a methodology that should serve as an alternative to evolving MO rules, whereas in [64] the ensembles are used to further improve the performance of evolved MO DRs.

3. Definition of the unrelated machines scheduling problem

The unrelated parallel machines environment is the most general single stage scheduling environment, in which each job needs to be processed only on a single machine before it is completed [1]. The main difference between this and other parallel machines environments (uniform and identical) is that in the unrelated machines environment a processing time is associated with each job-machine pair. These processing times usually have no relation with each other, such that one machine always executes jobs faster than another machine or a similar one. As such, this environment is more difficult to solve as no relations between jobs or machines can be extracted.

Formally, a problem in the unrelated machines scheduling environment consists of a finite number of jobs n and machines m. For each job j, a machine i that will execute that job needs to be selected. Each job can be assigned only to a single machine, and each machine can execute only a single job at any given moment in time. It is assumed that each job can be processed by each machine. After a job starts executing on a machine, it must be completed, i.e. it is not allowed to interrupt the execution of the job (no preemption is allowed). Each job j has several properties associated with it, such as:

- Processing time p_{ij} - the time required to process job j on machine i.
- Release time r_j - the time when job j is released in the system and can be scheduled.
- Due date d_j - the time until which job j should be executed or otherwise a certain penalty is invoked.
- Weight w_j - the importance of job j.

After the schedule is constructed, the completion time C_j can be calculated for each job j. Based on the previous properties, it is possible to define numerous scheduling criteria, from which the following will be considered in this study:

- C_{max} - maximum completion time of all jobs: $C_{max} = \max_j(C_j)$,

- Ft - total flowtime: $Ft = \sum_j F_j$, where $F_j = C_j - r_j$
- 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 which were executed on the machine with index i,
- Nwt - weighted number of tardy jobs: $Nwt = \sum_j w_j U_j$, where $U_j = \begin{cases} 1 : T_j > 0 \\ 0 : T_j = 0 \end{cases}$, and $T_j = \max(C_j - d_j, 0)$
- Twt - total weighted tardiness: $Twt = \sum_j w_j T_j$.

The reason why the C_{max}, Ft, Nwt, and Twt criteria were selected is since they are the most commonly considered criteria in the scheduling literature, both in SO and MO optimisation [64]. However, all those criteria are not largely conflicting, since by optimising each one of them the others will also be optimised to a certain extent. Therefore, the Mus criterion, which is used to distribute the load across the machines, is also included, since it demonstrated to be conflicting with the others to a much greater extent [35].

Based on the previous criteria, 8 MO scheduling problems will be considered in this study to evaluate the proposed method. The problems will be denoted using the standard $\alpha|\beta|\gamma$ classification scheme for scheduling problems [1]. In this notation, α denotes the considered machine environment (in this case denoted as R), β are the additional constraints included in the scheduling problem (in this case, the problem includes release times denoted as r_j), and γ is the set of optimised criteria (a combination of the five previously outlined scheduling criteria). The problems considered in this study are:

- $R|r_j|C_{max}, Twt$
- $R|r_j|C_{max}, Mus$
- $R|r_j|Ft, Twt$
- $R|r_j|C_{max}, Ft, Twt$

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

The reason why these MO problems were selected is to analyse the performance of the algorithms given different compositions of the problem they need to optimise, but also to analyse their performance on problem with various sizes. Therefore, the problems were constructed either to include the Mus criterion to determine how the method performs given that a highly conflicting criterion is included, or to exclude the criterion to analyse how the method performs on problems consisting only out of standard scheduling criteria. Furthermore, several of these MO problems were already considered in the literature in other studies [64].

The previously described scheduling problem is considered under dynamic conditions. This means that jobs are released into the system during its execution, as defined by their release times. However, no information about the job (not even its release time) is known before the job becomes available. Therefore, during the construction of the schedule it is not known when the next job would be released nor what its properties would be. Thus, it is not possible to create the entire schedule in advance, but rather it has to be constructed in parallel with the execution of the system. This makes DRs the most appropriate choice for solving such a problem.

4. Constructing ensembles of DRs for MO problems

This section provides the outline of the method for designing MO ensembles out of DRs evolved for optimising individual criteria. The first part of the section will describe how the individual rules are generated using GP, whereas the second part of the section outlines how they are combined into ensembles.

4.1. Designing DRs with GP

At its base, a DR is a simple greedy heuristic that constructs the solution incrementally, usually guided by a simple rule by which it selects the next job to be scheduled. The part of the DR that is concerned with determining when the scheduling decision should be performed, which decisions are eligible at a given moment and similar, is called the schedule generation scheme (SGS). On the other hand, the rule used to select the appropriate job to be scheduled next is called the priority function (PF). Therefore, when dealing with the design of a new DR, regardless of how this is done (manually or automatically), both parts of it need to be defined.

The SGS is usually the easiest part to define, as it follows a general pattern. It is always concerned with determining when a scheduling decision needs to be performed and ensuring that only feasible schedules are constructed (for example, that a job cannot be scheduled on a machine that is already executing another job). The outline of the SGS used in this study is presented in Algorithm 1. This SGS is used selected since a previous study demonstrated that it is superior to other alternative SGS variants used for the unrelated machines environment. As can be seen, the SGS is executed repeatedly during the execution of the system and at each decision point, when there is at least one released job and one available machine, the SGS determines the next job that will be scheduled. This is done by ranking all the job-machine pairs and based on a value obtained by a certain PF, selecting the pair with the best value, and scheduling the selected job on the corresponding machine.

From the previous description it is clear that the PF plays a vital role in the construction of the schedule. Many PFs were already designed manually for various criteria [17], however, the PFs used are usually quite simple and they perform decisions only based on a limited view of the problem. Therefore, the idea behind the automated design of DRs is to use a hyper-heuristic method to generate an adequate PF for the considered problem variant. Even though there are numerous hyper-heuristic methods proposed in the literature, this study uses genetic programming (GP) to design new PFs. GP is selected since

Algorithm 1 SGS used by generated DRs

while true **do**
    Wait until at least one job and machine are available
    for all available jobs j and each machine i in m **do**
        Calculate the priority π_{ij} of scheduling j on machine i
    end for
    for all available jobs **do**
        Determine the machine with the best π_{ij} value
    end for
    while jobs whose best machine is available exist **do**
        Determine the best priority of all such jobs
        Schedule the job with best priority
    end while
end while

it is the most commonly applied method in the literature for designing DRs [18], but also since it demonstrated to perform equally well as other alternative methods like artificial neural networks [65], or evolutionary computation methods like Cartesian GP or gene expression programming [66]. Since the PF is basically just an arithmetic expression, it may be encoded as an expression tree. Thus, GP is tasked with obtaining an appropriate expression for calculating priorities at different scheduling decisions. For that purpose, it is required to define the building blocks that will be used by GP when evolving new PFs. This set, denoted as the primitive set, consists out of various mathematical functions and terminal symbols. For function nodes, simple mathematical operators are used to construct the PF, such as addition, subtraction, multiplication, protected division (returns 1 when division by zero occurs), and the positive operator, which is a unary operator that returns the value of the argument if it is positive; otherwise, it returns zero. On the other hand, the set of terminal nodes is given in Table 1. These nodes represent various information about the system based on which the PF can efficiently rank the importance of individual scheduling deci-

Table 1: The set of terminal nodes used by GP to evolve PFs

Terminal	Description
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, $-max(d_j - p_{ij} - t, 0)$

sions. Both the function and terminal set were determined through exhaustive experiments [21].

The outline of the methodology used to generate new DRs for MO problems is given in Figure 1. For that purpose a MO metaheuristic, like NSGA-II or NSGA-III needs to be coupled with GP to evolve PFs for the given problem. This is done so that the MO algorithms simply use the expression tree representation of individuals and the corresponding crossover and mutation operators. Furthermore, the method also requires a training set, consisting of scheduling problem instances, on which it evaluates the quality the evolved individuals. After the evolution process finishes after reaching a certain termination criterion, the Pareto set of solutions is returned, which in this case represent individual PFs for the considered MO problems.

4.2. Constructing MO ensembles of DRs

As previously outlined, the main way in which DRs were evolved for solving MO problems was by using various MO algorithms in combination with GP to evolve such rules. For that sake, several MO algorithms have been successfully

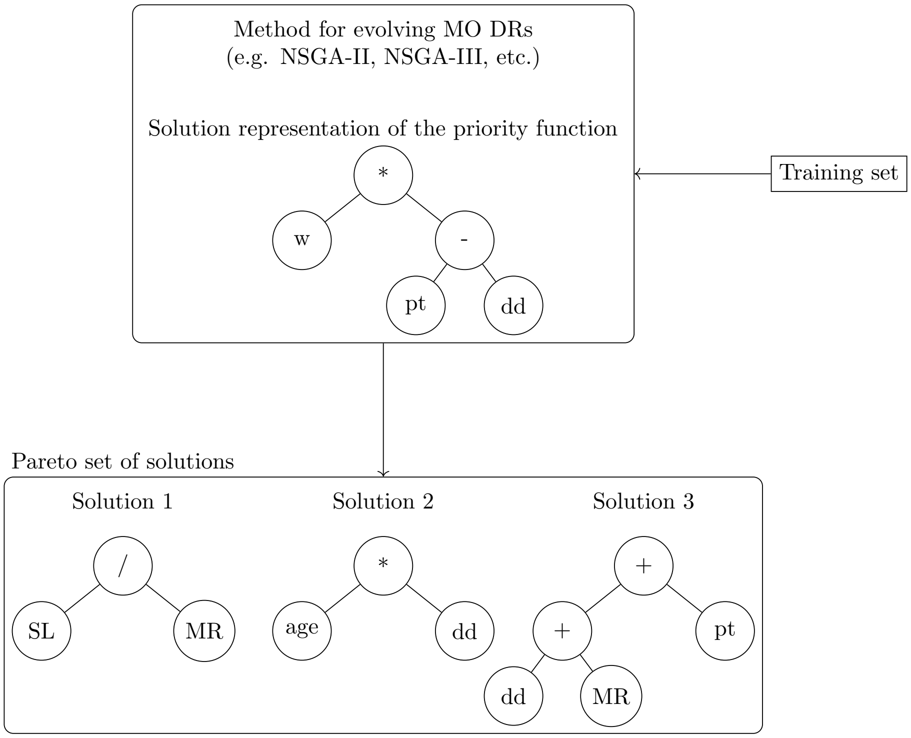


Figure 1: Flowchart outlining the evolution of new DRs for MO problems

applied, and the MO rules they evolved showed a better performance than existing manually designed rules [22, 35]. In all cases, it was required that new DRs are evolved from scratch, and no existing rules that had been previously evolved and performed well, can be reused. However, previous research indicated that evolved rules can be efficiently reused by forming ensembles which achieve a better performance than the individual rules [59, 37, 60, 67]. Based on these findings, we propose here a method based on ensemble learning to construct ensembles for MO problems out of DRs that were evolved for optimising individual criteria.

Since the task is to essentially construct an ensemble of DRs, four elements need to be defined to be able to construct ensembles:

1. The set of rules used to construct ensembles.

2. The size of the ensemble.
3. The ensemble combination method.
4. The ensemble construction method.

In the context of this problem, the set of rules used to construct the ensembles will consist of a given number of automatically designed DRs that were evolved previously using the GP hyper-heuristic described in the last section. Depending on which MO problem is considered, the set of rules will consist of rules that were evolved individually for each criterion contained in the MO problem. The details of constructing this set will be given in the experimental setup section. In general, the set of rules that is used can be constructed in an arbitrary way and can contain any number of rules larger than the size of the ensemble that is constructed.

The size of the ensemble defines the number of rules that will be included in the ensemble. The value of this parameter is quite important, as it directly influences the performance of the ensembles. Although different studies used various ensemble sizes, here we focus on smaller ensembles, with sizes of 3, 5, and 7. This is motivated by a previous study that showed that such ensembles already perform well enough, and that the increase in their size does not lead to significant improvement in the results [37].

The ensemble combination method defines how the rules in the ensemble will interact to perform a collective decision. Based on the results from previous studies [58] two combination methods will be used: sum and vote. In both cases, at each scheduling decision all rules in the ensemble are evaluated. If the sum method is used, then the priorities obtained by all rules for each job-machine pair are added, and the job-machine pair with the highest aggregated priority is selected. In the case of the vote combination method, each rule casts a single vote for the job-machine pair for which it obtained the best priority values. The votes of all rules are then aggregated for each job-machine pair, and the one that received the highest number of votes is selected. In case of ties, the job that was released first is selected.

The final thing that needs to be specified is the method that will be used to construct the ensembles. For the purpose of constructing ensembles, the previously proposed SEC method [37] is adopted for MO problems. The idea of this method is that it constructs a number of N ensembles by randomly sampling rules from a given set, which are then collected in an ensemble. In each iteration an ensemble is constructed and if it is better than the best one found until now, it is stored as the best one. Unfortunately, the original method is appropriate only for optimising single objective problems, and not for MO problems. However, this can be solved by introducing the notion of Pareto fronts and non-dominated sorting. Instead of storing only the current best solution in each iteration, all constructed ensembles are placed in a set upon which the non-dominated sort is performed at the end of the algorithm. Thus, all the obtained ensembles will be divided into several fronts based on the number of solutions that dominate them. The first front, i.e., the front of ensembles which are not dominated by any other ensemble, is then returned as the result of the SEC algorithm. This way, instead of providing only a single solution, the SEC method will provide a Pareto front of ensembles, and the appropriate ensemble can be selected by the user based on the trade-offs among different optimisation criteria.

Since finding good ensembles out of a given set of rules is basically an optimisation problem, we can also employ MO metaheuristics for that purpose. Therefore, we also apply the NSGA-II and NSGA-III algorithms, since they have shown a good performance in previous studies dealing with the design of DRs. In this case, the algorithms will use a simple integer chromosome with the size being equal to the size of the ensemble that needs to be constructed. The elements in the ensemble specify which rules are used to constitute the ensembles. A simple one-point crossover and flip-bit mutation (randomly changes) an element in the chromosome to a new value) are used during the evolutionary process. The above described genetic algorithm (GA) is quite simple. However, the purpose of the algorithm is to validate the performance of SEC in comparison to a GA that provides a certain guidance during the search. As SEC is already a very simple method, the goal was also to use a simple GA, although both methods could be further extended by introducing more sophisticated mechanisms to search the solution space.

The methodology used to construct MO ensembles from individual DRs is outlined in Figure 2. In this example, ensembles of size 3 are designed, meaning they consist out of 3 DRs. The main difference between this procedure, and the one described in the previous subsection, is that now PFs do not need to be evolved, but rather existing PFs previously evolved for SO problems are used and given as an input to the optimisation algorithm to construct ensembles. Based on this set the algorithm constructs ensembles, which can be done either by using SEC that simply selects rules and constructs and ensemble out of them. Since the selection DRs to form the ensemble is in fact an optimisation problem, MO optimisation algorithms like NSGA-II and NSGA-III can also be applied. However, instead of using an expression tree representation, they use a simple integer based representation that denotes which DR are used to form the ensemble. Since ensembles of size 3 are constructed, this means that in this case each solution will be represented as an array of size 3, where each element represents an index of one of the five previously evolved DRs. When the method terminates, it returns a Pareto set of integer arrays, which are decoded into ensembles using the set of evolved PFs and that are interpreted either by using the sum or vote combination method.

5. Experimental setup

In this section, we describe the setup of the experiments that were used to evaluate the performance of the proposed SEC method. The parameters considered for building ensembles are summarised in Table 2. The method was tested with three ensemble sizes of 3, 5, and 7, as well as with the sum and vote combination method. To further investigate the influence of some parameters, two stopping conditions were tested as well: 1 000 and 10 000 ensemble evaluations. Finally, the influence of using different number of DRs for constructing ensembles was also analysed. In this case, three sizes were tested: 10, 30, and 50 rules

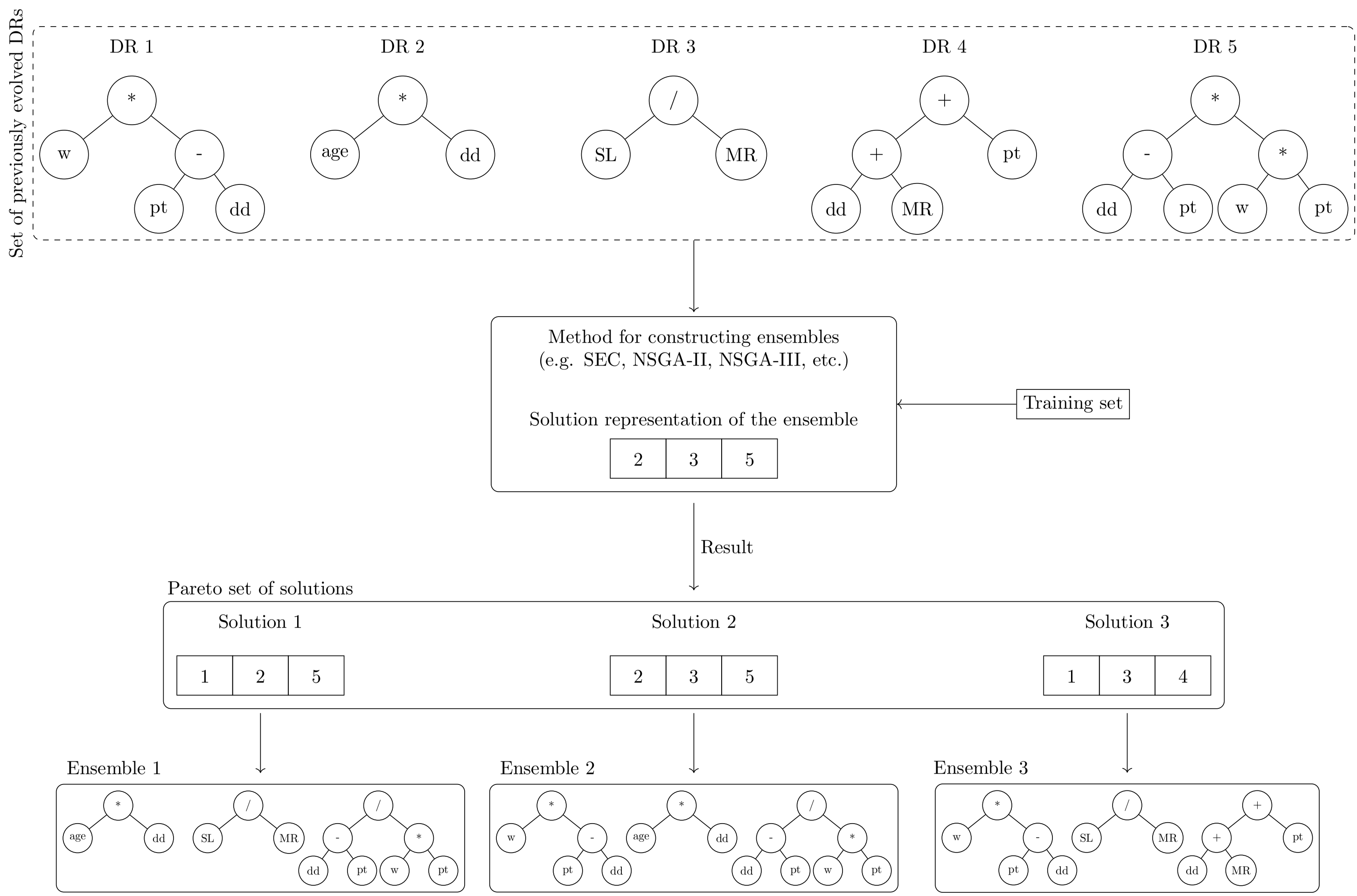


Figure 2: Flowchart outlining the evolution of ensembles of DRs for MO problems

per criterion. When selecting either 10 or 30 rules, they were sorted by their fitness on the training set, and the best 10 or 30 rules for each criterion were selected to be used for constructing ensembles. The DRs used to construct the ensembles were obtained in a previous study [35] with a standard GP algorithm, and the same set is used by all algorithms to construct ensembles.

To validate the effectiveness of SEC in constructing good ensembles, the NSGA-II and NSGA-III algorithms are also applied to generate ensembles of DRs. This means that both algorithms use an integer array representation of solutions, which denote the indices of DRs that are used to form the ensemble. The additional algorithm specific parameter values of these algorithms are outlined in Table 3, and were determined based on a preliminary set of experiments. The results of these algorithms will be denoted as E-NSGA-II and E-NSGA-III following sections, and are used to demonstrate that SEC can obtain Pareto fronts of solutions that are competitive to Pareto fronts of standard and more sophisticated evolutionary MO optimisation methods.

Finally, to have a better notion of how the constructed ensembles perform, they were compared against MO DRs evolved using NSGA-II and NSGA-III similar as done in [35]. The parameters used in these experiments are summarised in Table 4. Both algorithms used a population of 1 000 individuals and 100 000 function evaluations as the stopping criterion, whereas, for the mutation probability, NSGA-II used 0.1 while NSGA-III used 0.9. All these parameters were obtained through a preliminary tuning phase performed in a previous study [35]. It should be outlined that the NSGA-II and NSGA-III algorithms use more function evaluations than the ensemble construction methods. Although the two fitness functions are not exactly comparable, as in one evaluation of an ensemble more computation is performed than in an evaluation of a single rule, the algorithms that evolve the MO rules still have a longer execution time. As this number of function evaluations was found to produce the best results in the preliminary analysis, and the algorithms still execute longer than the proposed methods, we opted to keep such a stopping conditions to execute them using the best conditions obtained for them.

Table 2: Parameter values used in the experiments to construct ensembles.

Parameter name	Parameter values
Ensemble size	3, 5, and 7
Combination method	Sum and vote
Size set of rules	10, 30, and 50
Stopping condition	$1,000$, and 10 000 evaluations

Table 3: Parameter values used by E-NSGA-II, and E-NSGA-III algorithms.

Parameter name	Parameter value	
Population size	1 000	
Mutation probability	E-NSGA-II	0.1
	E-NSGA-III	0.9
Crossover operators	One point	
Mutation operators	Simple	
Stopping condition	10 000 evaluations	

Table 4: Parameter values used by NSGA-II and NSGA-III.

Parameter name	Parameter value	
Population size	1 000	
Mutation probability	NSGA-II	0.1
	NSGA-III	0.9
Crossover operators	Subtree, uniform, context-preserving, size-fair	
Mutation operators	Subtree, hoist, node complement, node replacement, permutation, shrink	
Stopping condition	100 000 evaluations	

In order to test the performance of the proposed methodology, a set of problem instances from previous studies is used [21]. This set is split into the training and the test set. The training set is used by SEC, E-NSGA-II, and E-NSGA-III to evolve ensembles, as well as by NSGA-II and NSGA-III to evolve DRs for optimising MO problems. Each of these methods returns a Pareto front of solutions, which either contains ensembles of DRs, or individual DRs. These Pareto fronts are then evaluated on the test set to obtain a notion of how the evolved rules and ensembles generalise well on unseen problem instances.

For comparing the Pareto fronts obtained by the different methods, we will use the hypervolume (HV) performance indicator. The reference point will be constructed using the worst values obtained across all the Pareto fronts obtained by each algorithm for the considered problem. Since each method was executed 30 times, the tables will outline the median value calculated based on the 30 HV values obtained for each execution. To test whether any statistical difference between the methods exists, the Kruskal Wallis test was used, followed by the post-hoc Dunn's test with the Bonferroni correction method. The differences in the result are considered significant if a p-value less than 0.05 is obtained.

6. Results

In this section, we denote the results obtained for the considered MO scheduling problems. The results are divided into subsections depending on the number of criteria that are optimised simultaneously.

6.1. Optimisation of two criteria

Table 5 outlines the results obtained for optimising the $R|r_j|C_{max}, Twt$ problem. The table outlines the median values of the HV obtained on the 30 executions for each experiment. Furthermore, each cell also outlines whether the results obtained by constructed ensembles are better than the results of MO DRs obtained by NSGA-II and NSGA-III. These results are denoted in brackets, in which the first element represents the result of the comparison with individual rules evolved by NSGA-II, whereas the second element represents the result of the comparison with individual rules evolved by NSGA-III. These elements can take values of + denoting that the ensembles achieved a better performance, ≈, denoting that ensembles and MO DRs perform equally well, and −, denoting that the ensembles performed significantly worse than MO DRs.

The results show that by constructing the ensembles out of SO rules it is possible not only to match, but even outperform the results obtained by evolving MO DRs with NSGA-II or NSGA-III. In most cases, ensembles significantly outperformed MO rules, and in the remaining few cases they performed equally well. What is interesting to observe is that SEC is able to match the performance of NSGA-II and NSGA-III by using only 10 rules per criterion and 1000 function evaluations. As the number of rules and evaluations increases, the ensembles constructed by SEC significantly outperform MO DRs evolved by NSGA-II and NSGA-III. By comparing the results between SEC and the two GAs for constructing ensembles, we see that there is no clear winner, and that both methods perform quite similarly. This suggests that there is little added value in the genetic operators for this problem, and that simple random sampling is already powerful enough to obtain good ensembles. Finally, regarding the size of the ensemble and combination method, we see that there are very few differences, although it seems that the sum combination method did lead to slightly better results.

Table 5: Results using the HV performance indicator for the $R|r_j|C_{max}, Twt$ problem

Method	NR	EVAL	sum			vote		
			3	5	7	3	5	7
SEC	10	1000	0.85 ($\approx$ | $\approx$)	0.85 ($\approx$ | $\approx$)	0.86 ($\approx$ | $\approx$)	0.82 ($\approx$ | $\approx$)	0.83 ($\approx$ | $\approx$)	0.84 ($\approx$ | $\approx$)
		10000	0.85 ($\approx$ | $\approx$)	0.86 (+|+)	0.87 (+|+)	0.84 ($\approx$ | $\approx$)	0.82 ($\approx$ | $\approx$)	0.85 ($\approx$ | $\approx$)
	30	1000	0.88 (+|+)	0.88 (+|+)	0.88 (+|+)	0.85 ($\approx$ | $\approx$)	0.88 (+|+)	0.88 (+|+)
		10000	0.87 (+|+)	0.88 (+|+)	0.89 (+|+)	0.86 (+| $\approx$)	0.88 (+|+)	0.88 (+|+)
	50	1000	0.88 (+|+)	0.88 (+|+)	0.86 (+|+)	0.85 ($\approx$ | $\approx$)	0.88 (+|+)	0.87 (+|+)
		10000	0.86 (+|+)	0.89 (+|+)	0.89 (+|+)	0.86 (+|+)	0.87 (+|+)	0.88 (+|+)
E-NSGA-II			0.85 ($\approx$ | $\approx$)	0.88 (+|+)	**0.90** (+|+)	0.86 (+| $\approx$)	0.86 (+|+)	0.86 (+|+)
E-NSGA-III			0.86 (+|+)	0.87 (+|+)	0.87 (+|+)	0.88 (+|+)	0.87 (+|+)	0.87 (+|+)
NSGA-II			0.80					
NSGA-III			0.82					

The results for the $R|r_j|C_{max}, Mus$ problem are summarised in Table 6. For this problem, we observe a completely different behaviour than for the previous problem. Namely, the proposed method barely manages to match the performance of NSGA-II and NSGA-III when using a smaller number of initial rules and a smaller number of iterations. As the number of rules and constructed ensembles increases, so does the HV of the obtained Pareto fronts, with no significant differences between SEC with NSGA-II and NSGA-III. It is interesting to note that using the sum combination usually leads to better results for ensembles, and that in most cases they achieve equally good results as MO rules. On the other hand, the vote combination method usually resulted in ensembles that performed significantly worse than MO DRs.

Table 6: Results using the HV performance indicator for the $R|r_j|C_{max}, Mus$ problem

Method	NR	EVAL	sum			vote		
			3	5	7	3	5	7
SEC	10	1000	0.76 $(-\|-)$	0.75 $(-\|-)$	0.75 $(-\|-)$	0.67 $(-\|-)$	0.71 $(-\|-)$	0.71 $(-\|-)$
		10000	0.76 $(-\|-)$	0.77 $(\approx\|-)$	0.76 $(-\|-)$	0.68 $(-\|-)$	0.72 $(-\|-)$	0.72 $(-\|-)$
	30	1000	0.76 $(\approx\|-)$	0.78 $(\approx\|-)$	0.77 $(\approx\|-)$	0.72 $(-\|-)$	0.73 $(-\|-)$	0.74 $(-\|-)$
		10000	0.79 $(\approx\|\approx)$	0.79 $(\approx\|\approx)$	0.79 $(\approx\|\approx)$	0.74 $(-\|-)$	0.76 $(-\|-)$	0.76 $(-\|-)$
	50	1000	0.79 $(\approx\|\approx)$	0.79 $(\approx\|\approx)$	0.79 $(\approx\|\approx)$	0.76 $(-\|-)$	0.77 $(\approx\|-)$	0.77 $(\approx\|-)$
		10000	0.79 $(\approx\|\approx)$	0.80 $(\approx\|\approx)$	0.80 $(\approx\|\approx)$	0.76 $(\approx\|-)$	0.78 $(\approx\|-)$	0.78 $(\approx\|-)$
E-NSGA-II			0.79 $(\approx\|\approx)$	0.80 $(\approx\|\approx)$	0.80 $(\approx\|\approx)$	0.77 $(\approx\|-)$	0.79 $(\approx\|\approx)$	0.79 $(\approx\|\approx)$
E-NSGA-III			0.79 $(\approx\|\approx)$	**0.81** $(\approx\|\approx)$	**0.81** $(\approx\|\approx)$	0.77 $(\approx\|-)$	**0.81** $(\approx\|\approx)$	0.80 $(\approx\|\approx)$
NSGA-II			0.79					
NSGA-III			**0.81**					

Table 7 outlines the HV values obtained for the $R|r_j|Ft, Twt$ problem. For this problem, the results are quite similar to those obtained for the $R|r_j|C_{max}, Twt$ problem. Again, there is either no significant difference between the results obtained with ensembles and with MO DRs, or ensembles perform better when a larger set of initial rules and a maximum number of evaluations are used. There is also no significant difference in the performance between SEC and E-NSGA-II or E-NSGA-III, which again indicates that both methods perform equally well. Finally, there seems to be no difference among the ensembles using the sum or vote combination method, nor among ensembles of different sizes. As such it seems that these parameters have a small effect on the results, at least for this problem.

Table 7: Results using the HV performance indicator for the $R|r_j|Ft, Twt$ problem

Method	NR	EVAL	sum			vote		
			3	5	7	3	5	7
SEC	10	1000	0.85 ($\approx$ | $\approx$)	0.83 ($\approx$ | $\approx$)	0.80 ($\approx$ | $\approx$)	0.80 ($\approx$ | $\approx$)	0.83 ($\approx$ | $\approx$)	0.82 ($\approx$ | $\approx$)
		10000	0.85 ($\approx$ | $\approx$)	0.84 ($\approx$ | $\approx$)	0.84 ($\approx$ | $\approx$)	0.81 ($\approx$ | $\approx$)	0.86 (+| $\approx$)	0.84 ($\approx$ | $\approx$)
	30	1000	0.87 (+|+)	0.86 (+| $\approx$)	0.86 (+| $\approx$)	0.86 (+| $\approx$)	0.85 ($\approx$ | $\approx$)	0.84 ($\approx$ | $\approx$)
		10000	0.90 (+|+)	0.88 (+|+)	0.87 (+|+)	0.87 (+|+)	0.86 (+| $\approx$)	0.85 ($\approx$ | $\approx$)
	50	1000	0.87 (+|+)	0.85 ($\approx$ | $\approx$)	0.85 ($\approx$ | $\approx$)	0.87 (+|+)	0.86 (+|+)	0.85 ($\approx$ | $\approx$)
		10000	0.90 (+|+)	0.87 (+|+)	0.86 (+| $\approx$)	0.88 (+|+)	0.87 (+|+)	0.87 (+|+)
E-NSGA-II			0.88 (+|+)	0.86 (+| $\approx$)	0.86 (+| $\approx$)	0.88 (+|+)	0.874 (+|+)	0.87 (+|+)
E-NSGA-III			**0.92** (+|+)	0.89 (+|+)	0.84 (+| $\approx$)	0.88 (+|+)	0.88 (+|+)	0.85 ($\approx$ | $\approx$)
NSGA-II			0.75					
NSGA-III			0.83					

6.2. Optimisation of three criteria

In this section the results for problems that include three scheduling criteria are analysed. The first problem under consideration is the $R|r_j|C_{max}, Ft, Twt$ problem, for which the obtained results are denoted in Table 8. The results show that even when optimising three criteria simultaneously the performance of ensembles is always equally good or better than that of MO DRs. However, this time the vote combination method performs slightly better, which is evident from the fact that for more parameter combinations significantly better results were achieved than by NSGA-II and NSGA-III. Again, the ensemble size does not seem to have a great influence on the performance, nor does the application of E-NSGA-II or E-NSGA-III.

Table 8: Results using the HV performance indicator for the $R|r_j|C_{max}, Ft, Twt$ problem

Method	NR	EVAL	sum			vote		
			3	5	7	3	5	7
SEC	10	1000	0.85 ($\approx$ | $\approx$)	0.84 ($\approx$ | $\approx$)	0.83 ($\approx$ | $\approx$)	0.85 ($\approx$ | $\approx$)	0.85 ($\approx$ | $\approx$)	0.85 ($\approx$ | $\approx$)
		10000	0.86 ($\approx$ | $\approx$)	0.85 ($\approx$ | $\approx$)	0.84 ($\approx$ | $\approx$)	0.85 ($\approx$ | $\approx$)	0.86 ($\approx$ | $\approx$)	0.86 (+| $\approx$)
	30	1000	0.88 (+|+)	0.86 (+| $\approx$)	0.85 ($\approx$ | $\approx$)	0.87 (+|+)	0.87 (+|+)	0.88 (+|+)
		10000	0.88 (+|+)	0.88 (+|+)	0.87 (+|+)	0.88 (+|+)	**0.89** (+|+)	**0.89** (+|+)
	50	1000	0.86 (+| $\approx$)	0.86 (+| $\approx$)	0.86 (+| $\approx$)	0.86 (+| $\approx$)	0.87 (+|+)	0.87 (+|+)
		10000	0.87 (+|+)	0.88 (+|+)	0.87 (+|+)	0.88 (+|+)	0.88 (+|+)	0.88 (+|+)
E-NSGA-II			0.86 (+| $\approx$)	0.87 (+|+)	0.86 (+| $\approx$)	0.87 (+|+)	0.88 (+|+)	0.88 (+|+)
E-NSGA-III			0.86 (+| $\approx$)	0.87 (+|+)	**0.89** (+|+)	0.87 (+|+)	0.87 (+|+)	0.86 (+| $\approx$)
NSGA-II			0.80					
NSGA-III			0.84					

The results for solving the $R|r_j|C_{max}, Nwt, Twt$ problem are outlined in Table 9. The obtained values show that a similar behaviour can be observed as for the previously considered problem, namely that the ensembles using the vote combination method usually perform better than those that use the sum combination method. However, for this problem such a behaviour is even more evident, since when using the sum combination method the ensembles were unable to outperform the MO DRs in most cases. On the other hand, with the vote combination method, the ensembles achieved significantly better results for all except one parameter value. This means that even for the smallest number of rules and iterations used, SEC was able to construct ensembles that outperform MO DRs. In addition, this time it seems that larger ensembles lead to slightly better results, although there is no significant difference between them.

Table 9: Results using the HV performance indicator for the $R|r_j|C_{max}, Nwt, Twt$ problem

Method	NR	EVAL	sum			vote		
			3	5	7	3	5	7
SEC	10	1000	0.67 ($\approx$ | $\approx$)	0.71 ($\approx$ | $\approx$)	0.68 ($\approx$ | $\approx$)	0.73 ($\approx$ | $\approx$)	0.76 (+|+)	0.77 (+|+)
		10000	0.72 ($\approx$ | $\approx$)	0.74 (+| $\approx$)	0.74 ($\approx$ | $\approx$)	0.77 (+|+)	0.78 (+|+)	0.79 (+|+)
	30	1000	0.74 ($\approx$ | $\approx$)	0.72 ($\approx$ | $\approx$)	0.73 ($\approx$ | $\approx$)	0.77 (+|+)	0.80 (+|+)	0.81 (+|+)
		10000	0.75 (+| $\approx$)	0.77 (+|+)	0.76 (+|+)	0.79 (+|+)	0.82 (+|+)	**0.83** (+|+)
	50	1000	0.72 ($\approx$ | $\approx$)	0.72 ($\approx$ | $\approx$)	0.69 ($\approx$ | $\approx$)	0.77 (+|+)	0.77 (+|+)	0.79 (+|+)
		10000	0.74 (+| $\approx$)	0.74 (+| $\approx$)	0.75 (+| $\approx$)	0.79 (+|+)	0.81 (+|+)	0.81 (+|+)
E-NSGA-II			0.72 ($\approx$ | $\approx$)	0.74 (+|+)	0.74 (+| $\approx$)	0.77 (+|+)	0.80 (+|+)	0.82 (+|+)
E-NSGA-III			0.73 ($\approx$ | $\approx$)	0.75 (+|+)	0.75 (+| $\approx$)	0.79 (+|+)	0.80 (+|+)	0.78 (+|+)
NSGA-II			0.63					
NSGA-III			0.70					

6.3. Optimisation of four criteria

In this section, we analyse the performance of the proposed methodology on two problems that include 4 optimisation criteria. Table 10 outlines the results obtained for the $R|r_j|C_{max}, Ft, Mus, Twt$ problem. The results obtained for this problem are similar to the results obtained when considering the $R|r_j|C_{max}, Mus$ problem, since in most cases the ensembles performed significantly worse than MO rules evolved by either NSGA-II or NSGA-III. Again, the vote combination method achieved significantly worse results than the sum method, which is consistent with the previous observation. For larger parameter values ensembles achieved results with no significant difference between them and the MO DRs, but only when using the sum combination method. One interesting thing that can be observed is that for this problem it seems to be more important to use a larger set of rules for constructing ensembles, rather than using a larger number of function evaluations. This can be seen from the fact that when using 50 rules per criterion to construct ensembles, even 1 000 function evaluations were enough to match the performance of MO DRs.

Table 10: Results using the HV performance indicator for the $R|r_j|C_{max}, Ft, Mus, Twt$ problem

Method	NR	EVAL	sum			vote		
			3	5	7	3	5	7
SEC	10	1000	0.71 (−|−)	0.70 (−|−)	0.69 (−|−)	0.59 (−|−)	0.60 (−|−)	0.59 (−|−)
		10000	0.72 (≈ | ≈)	0.72 (≈ |−)	0.71 (−|−)	0.62 (−|−)	0.64 (−|−)	0.64 (−|−)
	30	1000	0.71 (−|−)	0.71 (−|−)	0.71 (−|−)	0.58 (−|−)	0.60 (−|−)	0.61 (−|−)
		10000	0.74 (≈ | ≈)	0.74 (≈ | ≈)	0.74 (≈ | ≈)	0.65 (−|−)	0.66 (−|−)	0.66 (−|−)
	50	1000	0.73 (≈ | ≈)	0.73 (≈ | ≈)	0.73 (≈ | ≈)	0.63 (−|−)	0.64 (−|−)	0.63 (−|−)
		10000	0.76 (≈ | ≈)	0.76 (≈ | ≈)	0.76 (≈ | ≈)	0.67 (−|−)	0.68 (−|−)	0.68 (−|−)
E-NSGA-II			0.76 (≈ | ≈)	0.76 (≈ | ≈)	0.76 (≈ | ≈)	0.67 (−|−)	0.67 (−|−)	0.67 (−|−)
E-NSGA-III			0.76 (≈ | ≈)	0.76 (≈ | ≈)	0.76 (≈ | ≈)	0.67 (−|−)	0.70 (−|−)	0.70 (−|−)
NSGA-II			**0.77**					
NSGA-III			**0.77**					

The results for the $R|r_j|C_{max}, Ft, Mus, Twt$ problem are shown in Table 11. Interestingly, these results are quite similar to the results obtained for the $R|r_j|C_{max}, Nwt, Twt$ problem. For this problem, the vote combination method again performed much better than the sum combination method, since it managed to outperform MO DRs in most cases. However, ensembles using the sum combination method were rarely able to do so, and when they did, it was mostly by adopting an ensemble size of 3. This shows another trend, which seems that the sum combination method is more inclined towards using smaller ensembles, whereas the vote combination method performs usually better when using slightly larger ensembles. Again, the application of the E-NSGA-II and E-NSGA-III methods does not lead to any improvement in the results over the SEC method.

Table 11: Results for the HV metric for the $R|r_j|C_{max}, Ft, Nwt, Twt$ problem

Method	NR	EVAL	sum			vote		
			3	5	7	3	5	7
SEC	10	1000	0.67 ($\approx$ | $\approx$)	0.65 ($\approx$ | $\approx$)	0.63 ($\approx$ | $\approx$)	0.70 ($\approx$ | $\approx$)	0.71 (+|+)	0.72 (+|+)
		10000	0.75 (+|+)	0.70 ($\approx$ | $\approx$)	0.67 ($\approx$ | $\approx$)	0.73 (+|+)	0.74 (+|+)	0.74 (+|+)
	30	1000	0.69 ($\approx$ | $\approx$)	0.67 ($\approx$ | $\approx$)	0.66 ($\approx$ | $\approx$)	0.73 (+|+)	0.75 (+|+)	0.75 (+|+)
		10000	0.72 (+|+)	0.71 (+|+)	0.70 ($\approx$ | $\approx$)	0.76 (+|+)	0.78 (+|+)	**0.79** (+|+)
	50	1000	0.67 ($\approx$ | $\approx$)	0.66 ($\approx$ | $\approx$)	0.660 ($\approx$ | $\approx$)	0.72 (+|+)	0.73 (+|+)	0.73 (+|+)
		10000	0.71 (+|+)	0.70 ($\approx$ | $\approx$)	0.69 ($\approx$ | $\approx$)	0.76 (+|+)	0.77 (+|+)	0.77 (+|+)
E-NSGA-II			0.69 ($\approx$ | $\approx$)	0.69 ($\approx$ | $\approx$)	0.68 ($\approx$ | $\approx$)	0.75 (+|+)	0.75 (+|+)	0.75 (+|+)
E-NSGA-III			0.72 ($\approx$ | $\approx$)	0.67 ($\approx$ | $\approx$)	0.67 ($\approx$ | $\approx$)	0.74 (+|+)	0.77 (+|+)	0.77 (+|+)
NSGA-II			0.64					
NSGA-III			0.65					

6.4. Optimisation of five criteria

Finally, in this section we investigate the performance on a MO problem in which five scheduling criteria need to be optimised simultaneously. Table 12 shows the results obtained for the problem $R|r_j|C_{max}, Ft, Nwt, Mus, Twt$. In this case it is evident that the constructed ensembles for most of the tested parameters achieved significantly worse results. However, for the largest set of initial individuals and number of iterations, the ensembles could again match the performance of MO DRs. The vote combination method resulted in quite poor results overall, not being able to match the performance of MO DRs even once. On the other hand, given enough function evaluations, SEC was able to construct ensembles that perform as well as MO DRs when using the sum combination method. As such, it can be seen that for this kind of criterion, the parameters can significantly affect the performance of SEC, but with appropriately set parameter values, the ensembles can match the performance of the MO DRs.

Table 12: Results for the HV metric for the $R|r_j|C_{max}, Ft, Nwt, Mus, Twt$ problem

Method	NR	EVAL	sum			vote		
			3	5	7	3	5	7
SEC	10	1000	0.59 (−|−)	0.59 (−|−)	0.58 (−|−)	0.48 (−|−)	0.50 (−|−)	0.49 (−|−)
		10000	0.62 (≈ | ≈)	0.62 (≈ | ≈)	0.61 (− | ≈)	0.52 (−|−)	0.54 (−|−)	0.54 (−|−)
	30	1000	0.59 (−|−)	0.59 (−|−)	0.58 (−|−)	0.49 (−|−)	0.51 (−|−)	0.50 (−|−)
		10000	0.63 (≈ | ≈)	0.63 (≈ | ≈)	0.63 (≈ | ≈)	0.55 (−|−)	0.57 (−|−)	0.57 (−|−)
	50	1000	0.60 (−|−)	0.60 (−|−)	0.60 (−|−)	0.51 (−|−)	0.51 (−|−)	0.51 (−|−)
		10000	0.64 (≈ | ≈)	0.64 (≈ | ≈)	0.64 (≈ | ≈)	0.56 (−|−)	0.58 (−|−)	0.57 (−|−)
E-NSGA-II			0.63 (≈ | ≈)	0.64 (≈ | ≈)	0.64 (≈ | ≈)	0.56 (−|−)	0.58 (−|−)	0.57 (−|−)
E-NSGA-III			0.63 (≈ | ≈)	0.64 (≈ | ≈)	0.64 (≈ | ≈)	0.56 (−|−)	0.57 (−|−)	0.57 (−|−)
NSGA-II					**0.66**			
NSGA-III					0.65			

6.5. Graphical analysis of the results

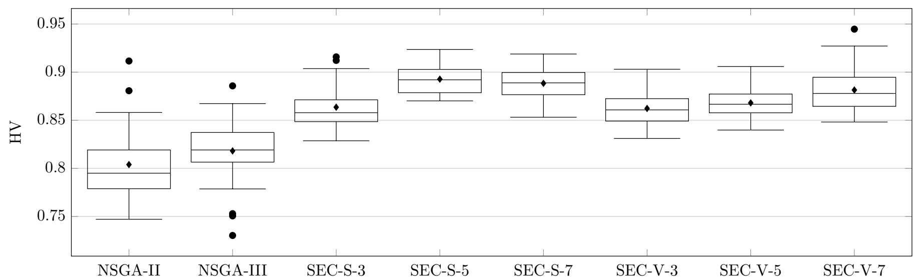
In order to gain a better notion of the difference in the performance of MO DRs and ensembles, we outline the box plots of the HV values for NSGA-II, NSGA-III, and SEC for all the considered problems. For SEC, we only include the results obtained when using 50 individuals per criterion and 10 000 ensemble evaluations, since for this configuration the method achieved the best overall results. The box plots are shown in Figure 3. For all the eight considered problems we can notice that the HV values obtained by the SEC method are less dispersed than those obtained both by NSGA-II and NSGA-III. This happens consistently across all problems, including those in which ensembles perform worse than MO rules or equally well. As such, we can conclude that SEC produces results that are, in general, less dispersed than the results obtained by NSGA-II and NSGA-III, which means that there is a greater chance of obtaining better Pareto fronts.

For all problems which do not include the Mus criterion, we see that the ensembles obtained much better distributions for the HV than rules evolved by either NSGA-II or NSGA-III. It is interesting to observe that for problems with a smaller number of optimised criteria both the sum and vote combination methods perform similarly. However, as the number of criteria in the problems increases, the ensembles using the vote combination method achieve better results than ensembles using the sum combination method. Thus, the vote combination method seems to scale better with the number of criteria that are optimised. When considering problems that include the Mus criterion, we notice a drastic change in the behaviour of the ensembles. In this case, the performance of ensembles using the vote combination method significantly deteriorates, and they perform significantly worse than ensembles using the sum combination method or MO DRs. On the other hand, the ensembles using the sum combination method still match the performance of MO DRs evolved by NSGA-II and NSGA-III. Thus, it seems that the sum combination method is more resistant to the composition of the MO problem than the vote combination method, although the vote combination method tends to be better in some cases.

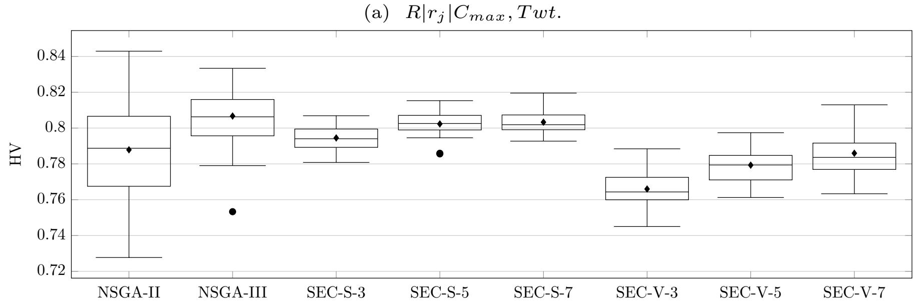
Finally, the effect of the ensemble size depends on the problem that was considered. In most cases, the ensemble size did not significantly affect the results, which means that even ensembles of size three performed well. However, in some cases, again most remarkably for problems that include the Mus criterion, using larger ensembles slightly improved the results. This is due to the fact that in those cases there is a higher probability of including rules that perform well for the Mus criterion, and provide better trade-offs between the optimised criteria and a better coverage of the objective space. But, this cannot be observed consistently, and the difference was usually noticed between ensembles of sizes 3 and 5. Therefore, it seems that the choice of the ensemble size is not as important, and that the SEC will be able to obtain good Pareto fronts regardless of the ensemble size adopted.

7. Further analysis

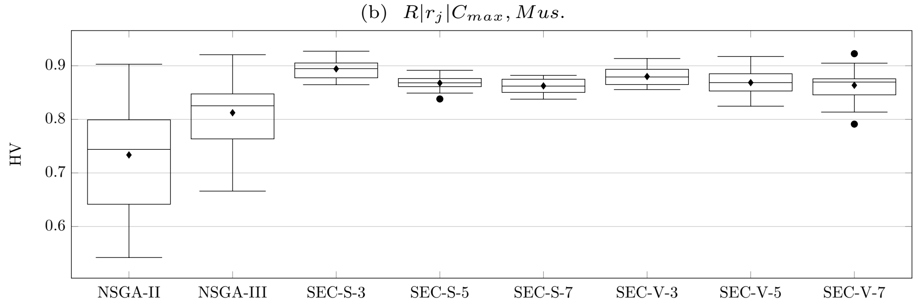
In this section, we perform several analyses and further investigations of the proposed methodology to gain a better understanding of its behaviour.



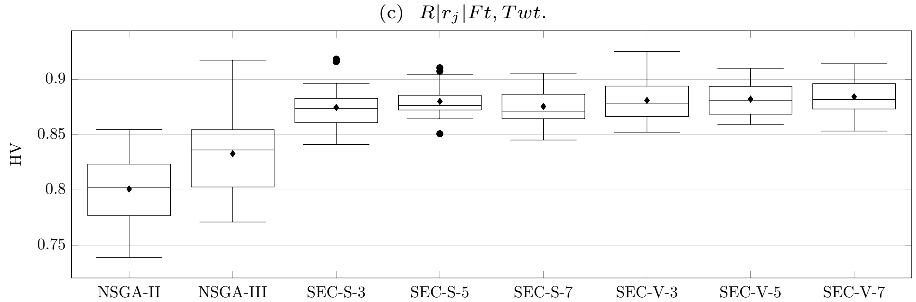
(a) $R|r_j|C_{max}, Twt.$



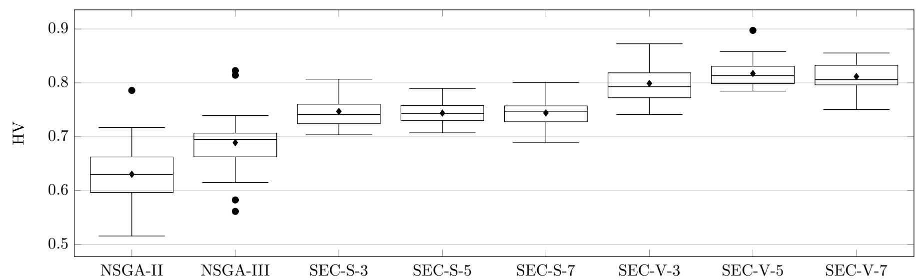
(b) $R|r_j|C_{max}, Mus.$



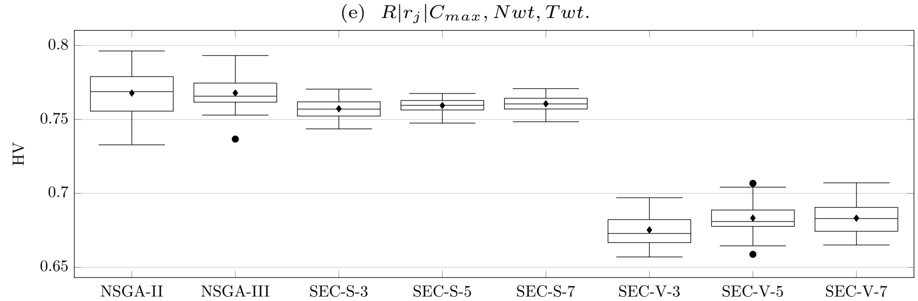
(c) $R|r_j|Ft, Twt.$



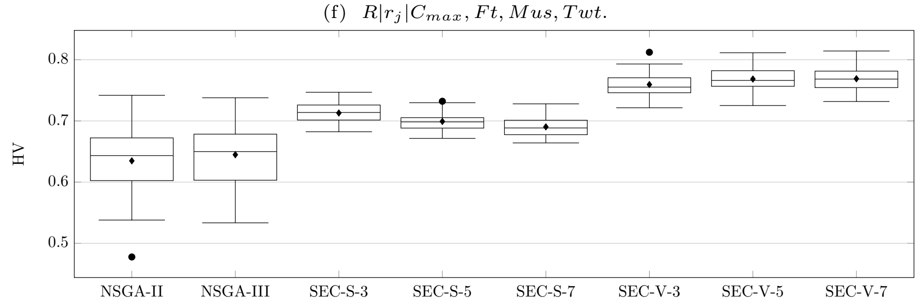
(d) $R|r_j|C_{max}, Ft, Twt.$



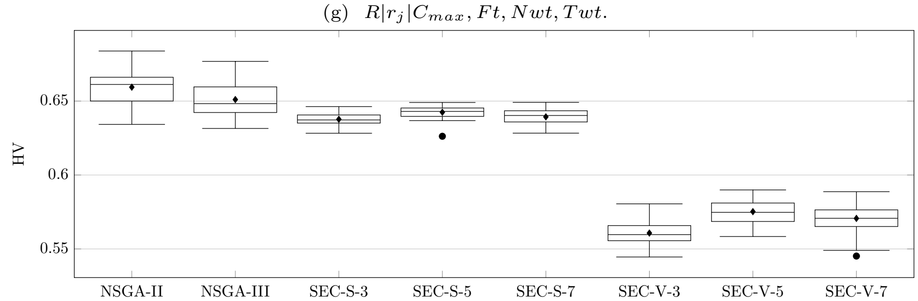
(e) $R|r_j|C_{max}, Nwt, Twt.$



(f) $R|r_j|C_{max}, Ft, Mus, Twt.$



(g) $R|r_j|C_{max}, Ft, Nwt, Twt.$



(h) $R|r_j|C_{max}, Ft, Mus, Nwt, Twt.$

Figure 3: Box plots of the HV values obtained by NSGA-II, NSGA-III, and SEC for the considered problems

7.1. Interaction between scheduling criteria

In the previous section we observed that for most of the considered problems the proposed methodology outperformed NSGA-II and NSGA-III in the quality of the obtained Pareto fronts. However, for certain problems we observed that there was no significant difference, or worse, that the Pareto fronts of MO DRs obtained by NSGA-II and NSGA-III performed significantly better. The problems on which the constructed ensembles performed worse were $R|r_j|C_{max}, Mus$, $R|r_j|C_{max}, Ft, Mus, Twt$, and $R|r_j|C_{max}, Ft, Nwt, Mus, Twt$. By examining all the problems, we can conclude that the only thing they have in common is that in each of them the Mus criterion is optimised, since this behaviour occurs regardless of the number of criteria that are optimised. As we know that this criterion is quite conflicting with other criteria, much more than all the others are mutually conflicting, this is the only logical explanation for such a behaviour. Therefore, in the rest of this section we try to gain a better understanding on how the five considered criteria affect each other.

The first thing we want to investigate is how the individual DRs evolved for one criterion perform on the other four scheduling criteria for which they were not optimised. Table 13 outlines the median values of the 50 rules evolved for one criterion (denoted in rows) on all five considered criteria (denoted in columns). What is immediately evident from the table is that rules evolved for optimising the Mus criterion perform poorly on all other criteria, as they achieve values several times larger than when those criteria were optimised. On the other hand, in the case when DRs are evolved for any of the other four criteria we can see a small degradation in the performance on the other criteria (except Mus), at most up to 30%. For example, when optimising the Twt criterion, a median of 13.60 was obtained. The rules obtained when optimising the C_{max} criterion performed worse than those rules by around 30% on the Twt criterion. However, the rules evolved for optimising the Mus criterion obtained a value of almost 690 for the Twt criterion, which is worse by a factor of 50 compared to the rules evolved for optimising the Twt.

As such, we can see a huge discrepancy in the performance of DRs across all

Table 13: Median values of evolved individual DRs across all five considered scheduling criteria

	C_{max}	Ft	Mus	Nwt	Twt
C_{max}	**38.26**	172.80	0.13	7.70	18.99
Ft	38.65	**155.01**	0.14	7.14	17.03
Mus	77.45	1521.44	**0.05**	42.24	687.12
Nwt	40.01	184.25	0.14	7.00	17.39
Twt	39.58	188.81	0.14	**6.69**	**13.60**

criteria depending on which criterion was optimised. The consequence of this is that it is much more difficult to construct ensembles for MO problems that include such conflicting objectives, as rules that perform well for one criterion will work poorly on another one, and vice versa. The problem could also be due to the fact that these SO rules only cover the extremes for these criteria, as they were optimised for each of them individually. As such there is not enough variety in the set of SO rules for SEC to use when constructing ensembles to cover the entire objective space well. A possible remedy would be to include more rules that perform not as well for each criterion in order to provide a better diversity of the rule set. Since for the other four criteria the differences are not that extreme, the constructed ensembles achieved great performance considering any combination of those criteria, which further backs up this hypothesis.

To further investigate the interaction among the different criteria, the Kendall rank correlation coefficient was calculated to determine the correlation between the criteria on which rules were optimised. Since for each criterion 50 DRs were evolved, these rules were also evaluated for the other criteria and the test was performed between the values obtained for the optimised criterion and each of the other four considered scheduling criteria. The results of these tests are outlined in Table 14, where rows denote which criterion was optimised, and columns the criteria with which the correlation was calculated. It can immediately be seen that in some cases there is a certain amount of correlation among the cri-

Table 14: Correlation values between different scheduling criteria

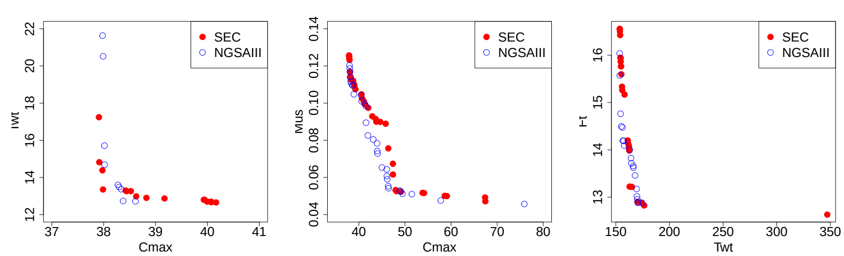
	C_{max}	Ft	Mus	Nwt	Twt
C_{max}		-0.23	0.32	-0.24	-0.22
Ft	0.42		0.28	0.07	0.43
Mus	-0.09	-0.14		-0.20	-0.13
Nwt	-0.02	-0.12	-0.09		0.16
Twt	0.03	0.08	-0.11	0.51	

teria. For example, the largest correlations exist between Twt and Nwt when optimising Twt, and between Ft and C_{max}, as well as Ft and Twt, when optimising Ft. However, it is interesting that this correlation is not bidirectional, which means that when C_{max} is optimised we observe even a slight negative correlation between it and Ft. Furthermore, the Mus criterion demonstrated to have no or slight negative correlation with all other criteria. It is also interesting to note that for the C_{max} criterion we observe that it had a slight negative correlation with all other criteria, except, surprisingly, for Mus. However, this had no effect on the performance of the ensembles when applied on problems that included this criterion. Thus, it seems that these correlation values cannot be used to determine whether the problem will be difficult or not to solve using ensembles. However, this value can possibly provide information on the problem and how it affects the ensembles, which will be discussed in a later section.

7.2. Pareto front visualisation

To gain a better understanding of the coverage of the Pareto fronts obtained by individual DRs and ensembles, this section provides visualisations of the Pareto fronts for the considered problems. The outlined Pareto fronts denote the union of the 30 Pareto fronts obtained in each execution of the algorithm. As for problems which include three or more criteria it is more difficult to visualise the Pareto fronts, we outline all pairwise combinations between the criteria for each of them. To make the figures more readable, we outline the Pareto fronts only for the NSGA-III algorithm (as it usually performed better) and the SEC method that uses 50 rules and 10000 iterations, but with the best obtained configuration for the size and ensemble collaboration method for each problem.

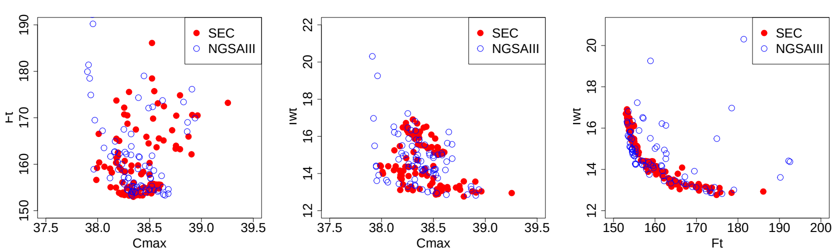
The Pareto fronts for the problems in which two criteria were optimised simultaneously are presented in Figure 4. For the problems $R|r_j|C_{max}, Twt$ and $R|r_j|Ft, Twt$ it is difficult to visually assess which Pareto front would be better, since both provide a good coverage of the objective space, but neither is consistently better. In some places, the MO DRs provide a better trade-off between the criteria and in other places the ensembles are better. However, for the $R|r_j|C_{max}, Mus$ problem it is evident that the MO DRs have a better convergence in the middle of the objective space, which translates into rules that provide a trade-off between the two criteria, whereas on the extremes for each criterion, the quality seems to be similar. As outlined in the previous section, this is probably the consequence of having no SO rules that cover that space by themselves, and as such SEC does not have the required material to construct ensembles that provide a good performance on that part of the objective space. On the other hand, for the $R|r_j|C_{max}, Twt$ it is interesting to observe that rules and ensembles that achieve the best performance for the C_{max} criterion perform equally well for that criterion, but the ensembles perform also much better for the Twt criterion. Therefore, in that case, ensembles are more efficient as they provide a better performance on the second criterion compared to MO DRs.



(a) $R|r_j|C_{max}, Twt$. (b) $R|r_j|C_{max}, Mus$. (c) $R|r_j|Ft, Twt$.

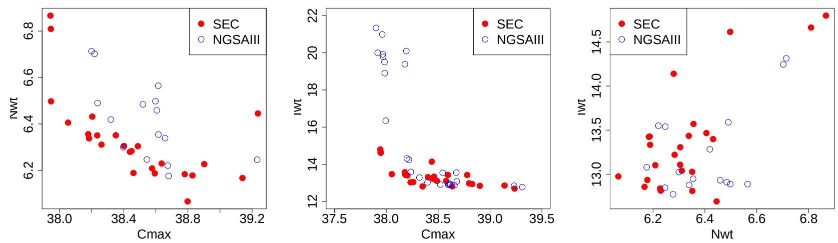
Figure 4: The Pareto front obtained for optimising problems with two criteria.

Figures 5 and 6 outline the Pareto fronts obtained when optimising the $R|r_j|C_{max}$, Ft, Twt and $R|r_j|C_{max}$, Nwt, Twt problems. In both cases, but especially for the second problem, it is evident that the ensembles provide a slightly better convergence and coverage of the objective space compared to MO DRs. This can best be seen in Figure 5c, where with SEC, a nice and compact distribution of solutions is obtained, whereas those obtained by MO DRs are more dispersed and do not provide an equally good coverage. Also, Figure 6a shows that for this combination of criteria the ensembles provide a much better convergence in comparison to MO DRs, but this can also be seen on the other plots although not as clearly. Naturally, there are cases when MO DRs provide a better convergence in parts of the objective space, as can be seen in Figure 5b. However, this usually happens only for a part of the objective space, but not across the entire objective space. Therefore, we see that, as the number of criteria increases, SEC can still provide a good coverage in the objective space for all criteria.



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

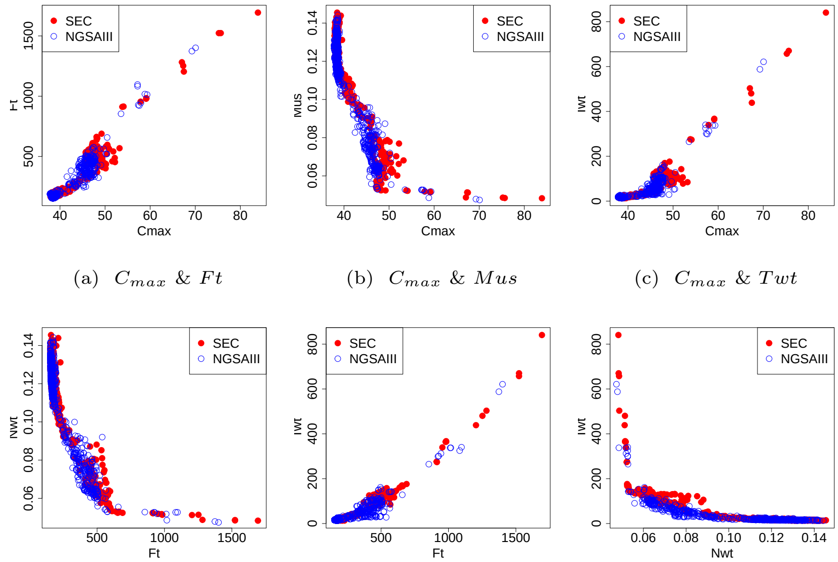
Figure 5: The Pareto front obtained for the $R|r_j|C_{max}$, Ft, Twt problem denoted through pairwise combinations of the three optimised criteria Pareto front.



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

Figure 6: The Pareto front obtained for the $R|r_j|C_{max}$, Nwt, Twt problem denoted through pairwise combinations of the three optimised criteria Pareto front.

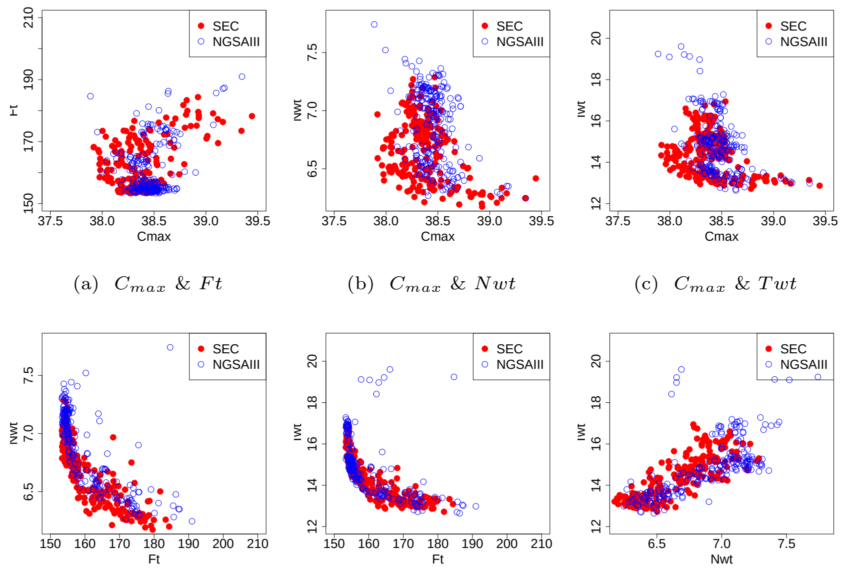
Figures 7 and 8 outline the Pareto fronts obtained for the problems that include 4 scheduling criteria, namely the $R|r_j|C_{max}$, Ft, Mus, Twt and $R|r_j|C_{max}$, Ft, Nwt, Twt problems. For the first problem denoted in Figure 7 we can observe a quite similar distribution of Pareto fronts obtained between ensembles and MO DRs. However, it seems that MO DRs even achieve a slightly better convergence. But this is not surprising since this problem includes the Mus criterion, which means that the ensembles will not perform as well on this problem as on some others. On the other hand, for the $R|r_j|C_{max}$, Ft, Nwt, Twt problem, we again see that with ensembles a better convergence and coverage of the objective space was obtained compared with MO DRs. This can be seen consistently for all pairs of criteria, although in some cases like in Figure 8f MO DRs also provide a good coverage, it is still not as good as the one obtained by ensembles.



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

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

Figure 7: The Pareto front obtained for the $R|r_j|C_{max}$, Ft, Mus, Twt problem denoted through pairwise combinations of the four optimised criteria Pareto front.

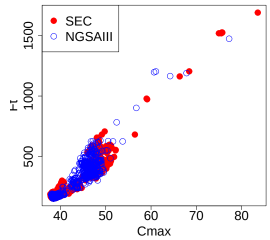


(a) C_{max} & Ft (b) C_{max} & Nwt (c) C_{max} & Twt

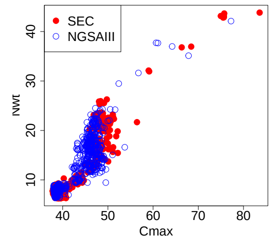
(d) Ft & Nwt (e) Ft & Twt (f) Nwt & Twt

Figure 8: The Pareto front obtained for the $R|r_j|C_{max}$, Ft, Nwt, Twt problem denoted through pairwise combinations of the four optimised criteria Pareto front.

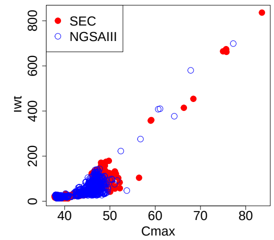
Finally, the Pareto fronts for the largest considered problem are shown in Figure 9. Due to the large number of solutions, it is difficult to see the differences between the Pareto fronts obtained by NSGA-III and SEC. However, they seem to follow a very similar pattern. Again they cover the same regions, and there does not seem to be a significant difference in the coverage or convergence of the Pareto fronts, although it does seem that MO DRs in some cases result in solutions that provide a better convergence. However, this is difficult to assess this due to the large number of points and large distributions of solutions for all criteria.



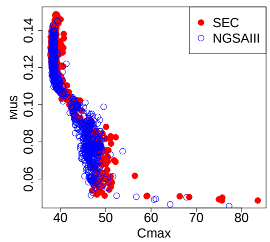
(a) C_{max} & Ft



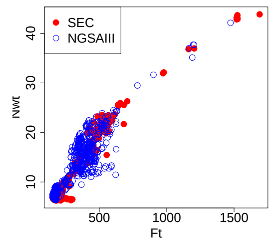
(b) C_{max} & Nwt



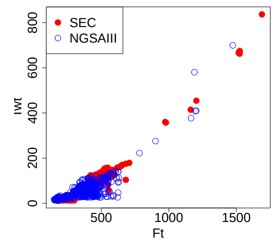
(c) C_{max} & Twt



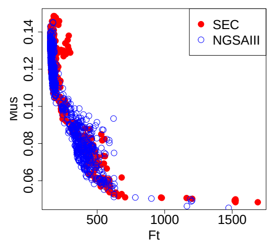
(d) C_{max} & Mus



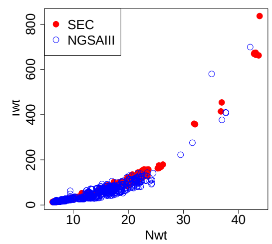
(e) Ft & Nwt



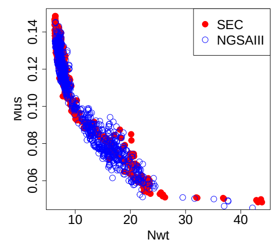
(f) Ft & Twt



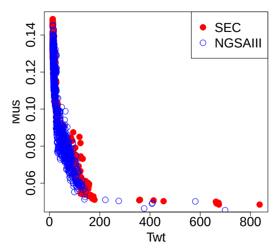
(g) Ft & Mus



(h) Nwt & Twt



(i) Nwt & Mus



(j) Twt & Mus

Figure 9: The Pareto front obtained for the $R|r_j|C_{max}$, Ft, Nwt, Twt, Mus problem denoted through pairwise combinations of the five optimised criteria Pareto front.

7.3. Generalisation capability of rules and ensembles

In this section, we want to further investigate the generalisation ability of the evolved rules and ensembles. However, we are interested in a different type of generalisation than the one tested by evaluating the rules and ensembles on unseen problem instances. In this case, the evolved DRs and ensembles will be again evaluated on the test set. However, we will now adopt problems that include a different number of criteria (both larger and smaller) than the original problem on which they were evolved. With this analysis we can obtain a notion of whether it is possible to reuse rules and ensembles evolved for another problem with a slightly different set of criteria, and what performance implications it would have.

Table 15 shows the results of the HV measure in the case when the Pareto front of rules or ensembles constructed for a problem with a larger number of criteria is applied on a problem with a smaller number of criteria. The column denoted as ‘original’ denotes the set of criteria for which the rules/ensembles were evolved, whereas the ‘reduced’ column denotes the set of criteria on which the obtained Pareto front was evaluated. For brevity, the results for only 4 methods are presented in the table, since the results for other parameter combinations are quite similar. The methods are outlined in columns, with NSGA-II, NSGA-III, SEC-S-3, and SEC-V-7 representing the results obtained when directly optimising the criteria in the ‘reduced’ column, while the methods with the prefix ‘R-’ represent the results obtained by rules and ensembles evolved for the criterion in the ‘original’ column, and then evaluated on the criterion in the ‘reduced’ column. For SEC, the combination method (S for sum, and V for vote), and the size of the constructed ensemble (3 or 7) are also denoted in the name of the method. These parameters were selected to examine both combination methods and both a larger and smaller ensemble size. Furthermore, to test the difference between the results obtained by optimising the reduced problem directly, and using a result from a larger problem, the Mann-Whitney statistical test was performed to determine whether there is a significant difference between the obtained HV values. These results are denoted beside the results outlined for "R-" methods and represent a comparison with the results of the statistical test with the corresponding method applied directly on the reduced problem set, i.e. the method with the same name but without the $R-$ prefix.

The results outlined in the table show something very interesting. Both, the rules and the ensembles that had been evolved on a larger set of criteria can efficiently be applied on a reduced set of criteria, with the HV usually staying the same or even increasing. The most interesting case is when rules/ensembles evolved for the largest problem (containing five criteria) are applied for smaller criteria sets (even for only two criteria), since the results are comparable and even in many cases better to those obtained by directly optimising the reduced problems. The same can be observed even if MO DRs were evolved on a larger set of criteria, and then evaluated on a smaller problem. This result has some quite interesting implications, as it shows that Pareto fronts evolved for larger criteria sets have an additional level of generalisation ability, and can be efficiently applied on problems with a subset of criteria. This would mean that it would be sufficient to evolve a Pareto front for a problem with a larger set of criteria, and that the solutions obtained within this Pareto front could be efficiently applied on subsets of this MO problem. As a consequence, it is not required to evolve Pareto fronts for each possible combination of criteria, as is the case when directly solving MO problems instead of developing heuristics. In that case, we can produce a Pareto front only once and then reuse the solutions obtained in that Pareto front.

Looking at the results of statistical tests, we see that when using rules evolved for larger problems to solve smaller problem sets, in 14 cases they can outperform the results of rules obtained by directly optimising the reduced problem, whereas in 14 cases they perform equally well and in 6 cases they achieve inferior results. On the other hand, ensembles evolved for larger problems and applied on smaller problems achieve a better performance than ensembles evolved directly for the reduced problem in 29 cases, an equal performance in 3 and in the remaining 2 cases they perform significantly worse. Based on these results, we can conclude that the strategy of applying a Pareto set of solutions obtained for

Table 15: Results of ensembles and rules when applying them on problems with a smaller number of criteria than the ones for which they were originally evolved

Original	Reduced	NSGA-II	R-NSGA-II	NSGA-III	R-NSGA-III	SEC-S-3	R-SEC-S-3	SEC-V-7	R-SEC-V-7
$C_{max}, Ft, Mus, Nwt, Twt$	C_{max}, Ft, Mus, Twt	0.740	**0.743**≈	0.737	**0.734**≈	0.729	0.728≈	0.660	0.667≈
	C_{max}, Ft, Nwt, Twt	0.659	0.608−	0.664	0.582−	0.726	0.735+	0.772	**0.791**+
	C_{max}, Ft, Twt	0.791	0.803≈	0.821	0.793−	0.858	0.881+	0.866	**0.892**+
	C_{max}, Nwt, Twt	0.662	0.657≈	0.712	0.623−	0.759	0.789+	0.814	**0.841**+
	C_{max}, Mus	0.795	0.808≈	**0.813**	0.801−	0.803	0.811+	0.796	0.780−
	C_{max}, Twt	0.781	0.808+	0.803	0.796≈	0.841	0.893+	0.861	**0.900**+
C_{max}, Ft, Mus, Twt	C_{max}, Ft, Twt	0.779	0.776≈	0.810	0.804≈	0.846	0.864+	0.854	**0.867**+
	C_{max}, Mus	0.799	0.805≈	0.816	0.807−	0.806	**0.821**+	0.800	0.785−
	C_{max}, Twt	0.769	0.777≈	0.791	0.809+	0.828	**0.877**+	0.848	0.875+
	Ft, Twt	0.731	0.758≈	0.809	0.797≈	0.878	**0.893**+	0.853	0.882+
C_{max}, Ft, Nwt, Twt	C_{max}, Ft, Twt	0.779	0.799+	0.810	0.810≈	0.846	0.857+	0.854	**0.869**+
	C_{max}, Nwt, Twt	0.652	0.695+	0.702	0.697≈	0.748	0.767+	0.802	**0.806**≈
	C_{max}, Twt	0.769	0.799+	0.791	0.811+	0.828	0.866+	0.848	**0.872**+
	Ft, Twt	0.731	0.825+	0.809	0.844+	0.878	0.894+	0.853	**0.904**+
C_{max}, Ft, Twt	C_{max}, Twt	0.769	0.778+	0.791	0.810+	0.828	0.854+	0.848	**0.857**+
	Ft, Twt	0.731	0.765+	0.809	0.842+	0.878	**0.897**+	0.853	0.890+
C_{max}, Nwt, Twt	C_{max}, Twt	0.769	0.785+	0.791	0.819+	0.828	0.862+	0.848	**0.866**+

larger problems on smaller problems is not only feasible, but can actually lead to improved results in many cases. And although this is the case both for Pareto sets of individual rules and ensembles, the results demonstrate that performing this strategy on ensembles is preferred, as in almost all cases significantly better results were achieved. Thus, we can assume that the Pareto sets obtained by ensembles of individual DRs are much more adaptable to other problem variants than individual rules.

Table 16 outlines the results of the second experiment, in which the generalisation ability of a Pareto front was tested when trying to apply it for a problem with a larger criteria set than the one for which the rules and ensembles were originally evolved. The outline of the table is the same as the previous one, with the first column 'original' denoting the original problem on which rules and ensembles were evolved, and the second column 'increased' denotes the problem with a larger criteria set on which they were evaluated. The selected methods are the same as in the previous experiment, with the only exception that the results for Pareto fronts that are evaluated on larger problems are now denoted with the prefix 'I-'. The Mann-Whitney test was used again to determine whether there is a statistically significant difference between the results obtained when directly optimising the larger problem and when using a Pareto set of solutions obtained when optimising a smaller problem.

In this case, we can observe two distinct patterns in the results. For some problems, the HV is reduced slightly when testing the Pareto front on the larger criteria set, whereas for some problems a huge decrease in the performance can be observed (the HV is reduced by a factor of 2 or more). By careful observation we can see that the dramatic decrease in the performance happens when the original problem did not include the Mus criterion, but the increased set did. This is again a consequence of this criterion being strongly conflicted with all other criteria. As such, if the original problem on which the Pareto fronts were evolved did not include this criterion, the obtained Pareto fronts simply provide a poor coverage for it. Since this criterion is conflicting with the others, it means that these Pareto fronts will cover only a small portion of the objective space, and thus obtain quite poor results. However, in all other cases, it can be said that it is possible to apply the evolved Pareto fronts also on problems that include more criteria, however, with a certain penalty on performance. Naturally, better performance is achieved if the problem is extended by only one criterion, but it is also possible to extend the problem with two criteria, although with a larger performance penalty.

The statistical results demonstrate that this strategy does not perform equally well as the previous one. In most cases the results obtained by using a Pareto set of solutions obtained for a smaller problem are significantly worse than those obtained by directly optimising the larger problem. However, these results are somewhat expected, as the methods did not focus on some of the objectives in the larger problems during the optimisation process, and as such had no possibility to search for those solutions that would perform well on them.

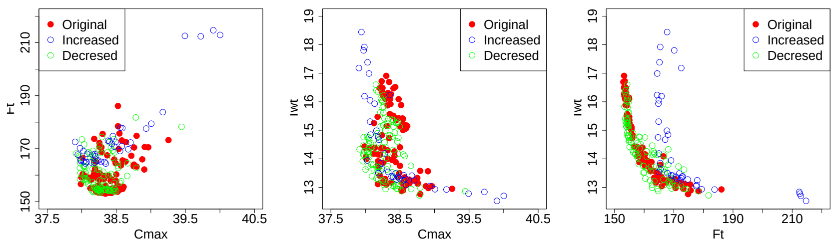
In addition to the previous results, we also graphically outline the Pareto fronts of three selected problems in Figures 10, 11 and 12. These figures outline the results obtained by SEC optimised for the considered criteria (denoted in the figure as 'original'), but also by a selected Pareto front obtained for a larger problem and applied for this problem (denoted in the figure as 'decreased'), and a selected Pareto front obtained by SEC for a smaller problem, which was then applied for this larger problem (denoted in the figure as 'increased'). The best Pareto fronts were selected for each of the three configurations.

Figure 10 outlines the Pareto fronts for the criterion consisting of three objectives. It is immediately clear that the Pareto fronts obtained by the original and the decreased SEC cover a similar area of the objective space. In some cases the decreased version also obtains better ensembles, thus it has a slightly better convergence, which can best be seen in Figure 10b. However, the results of the increased version are scattered all over the objective space. Especially from Figure 10c it is quite evident that for the Ft criterion the results obtained by these ensembles are quite poor. However, since the original problem for which these ensembles were evolved did not include this criterion, such a behaviour is expected. Nevertheless, this explains why its performance is not on par with

Table 16: Results of ensembles and rules when applying them to problems with a larger number of criteria than the ones for which they were originally evolved

Original	Increased	NSGA-II	I-NSGA-II	NSGA-III	I-NSGA-III	SEC-S-3	I-SEC-S-3	SEC-V-7	I-SEC-V-7
C_{max}, Ft, Mus, Twt	$C_{max}, Ft, Mus, Nwt, Twt$	**0.663**	0.661≈	0.651	0.655≈	0.629	0.641+	0.562	0.552−
C_{max}, Ft, Nwt, Twt	$C_{max}, Ft, Mus, Nwt, Twt$	**0.663**	0.250−	0.651	0.247−	0.639	0.241−	0.575	0.232−
C_{max}, Ft, Twt	$C_{max}, Ft, Mus, Nwt, Twt$	**0.663**	0.235−	0.650	0.242−	0.639	0.221−	0.574	0.214−
	C_{max}, Ft, Nwt, Twt	0.657	0.590−	0.663	0.638≈	0.723	0.708−	**0.768**	0.732−
	C_{max}, Ft, Mus, Twt	**0.750**	0.231−	0.747	0.238−	0.742	0.215−	0.676	0.208−
C_{max}, Nwt, Twt	C_{max}, Ft, Nwt, Twt	0.657	0.603−	0.663	0.656≈	0.723	0.692−	**0.768**	0.737−
	$C_{max}, Ft, Mus, Nwt, Twt$	**0.663**	0.241−	0.650	0.254−	0.639	0.241−	0.574	0.231−
C_{max}, Twt	C_{max}, Ft, Twt	0.779	0.727−	0.810	0.759−	0.846	0.798−	**0.854**	0.808−
	C_{max}, Ft, Nwt, Twt	0.671	0.558−	0.677	0.594−	0.740	0.654−	**0.790**	0.695−

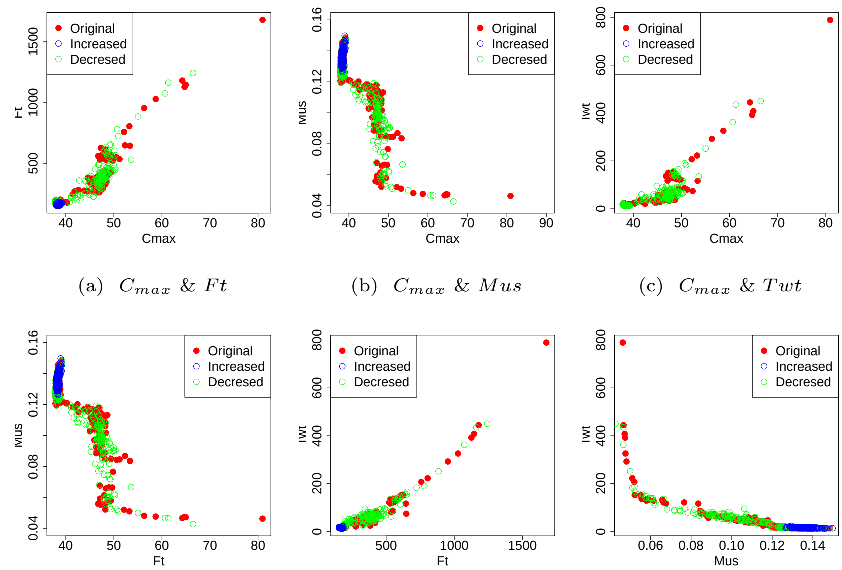
the other two SEC variants.



(a) $R|r_j|C_{max}, Ft.$ (b) $R|r_j|C_{max}, Twt.$ (c) $R|r_j|Ft, Twt.$

Figure 10: The Pareto front obtained for the $R|r_j|C_{max}$, Ft, Twt problem denoted through pairwise combinations of the three optimised criteria Pareto front.

Figures 11 and 12 outline the Pareto fronts obtained for problems with four criteria. A very interesting result can be observed for the $R|r_j|C_{max}$, Ft, Mus, Twt problem. In this case, the ensembles of the increased SEC variant were evolved on a problem that did not include the Mus criterion. It can be immediately seen that because of this, these ensembles have a poor coverage of the objective space. This is due to the fact that was previously outlined: this criterion is highly conflicting with all others, and thus if not optimised, the ensembles will only cover a very small range of the objective space in which they optimise the other criteria. As such, these ensembles only cover the search space for the other three criteria, and the part on which they perform well. On the other hand, in the Pareto fronts for the original and decreased SEC ensembles we observe a similar coverage of the search space, and there does not seem to be a significant difference between them.

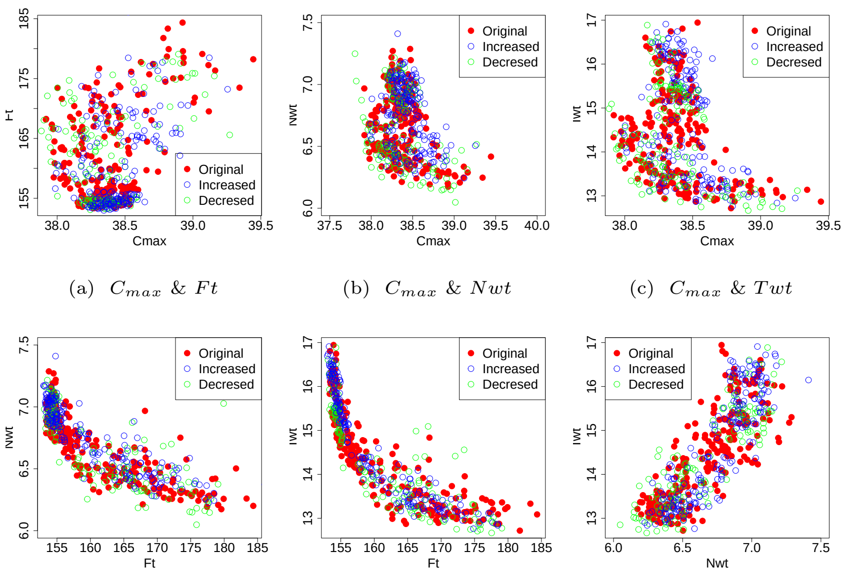


(a) C_{max} & Ft (b) C_{max} & Mus (c) C_{max} & Twt

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

Figure 11: The Pareto front obtained for the $R|r_j|C_{max}$, Ft, Mus, Twt problem denoted through pairwise combinations of the four optimised criteria Pareto front.

On the other hand, for the Pareto fronts in Figure 12, we observe that, since the Mus criterion is not considered, even the increased SEC version obtains a good coverage of the objective space. However, this Pareto front is still inferior when compared to the other two Pareto fronts, as we can notice that there are parts in the search space that are poorly covered by it. In this case, the original problem did not include the Nwt criterion, the consequences of which can be seen from the fact that it provides a very poor convergence for it, which is evident from Figures 12b, 12d, and 12f. For the other two Pareto fronts, neither can be said to be better, as they both seem to cover the search space well. But, in general, it does seem like the original SEC ensembles provide a better coverage, while on the other hand, the decreased ensembles are sometimes better in their performance.



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

(d) Nwt & Ft (e) Ft & Twt (f) Nwt & Twt

Figure 12: The Pareto front obtained for the $R|r_j|C_{max}$, Ft, Nwt, Twt problem denoted through pairwise combinations of the four optimised criteria Pareto front.

7.4. Extending ensembles

The analysis in the previous section outlined that although it is possible to use ensembles evolved for problems with more criteria on problems with fewer criteria, it is not very efficient to use ensembles evolved on problems with a few criteria on problems with a larger number of criteria. However, ensembles can be easily extended by adding additional rules into the ensemble. Thus, we want to analyse whether by adding additional rules in the ensembles it is possible to improve the performance of ensembles constructed for smaller problems, but in a shorter time than constructing them from scratch. For that purpose, ensembles of size 3 constructed for a smaller problem were extended to sizes 5 and 7, and ensembles of size 5 were extended to ensembles of size 7. The extension was done in a way that in each iteration of SEC a random ensemble was selected from the Pareto front evolved for the smaller criterion. The rules already contained in the ensemble were kept fixed, and the method added additional rules until the desired ensemble size was reached.

The results of this analysis are summarised in Table 17. Each column represents one experiment in which from the original ensemble evolved for the criterion denoted in the first row was incremented for the larger problem denoted in the second row. Each row represents the results obtained by SEC ensembles constructed for the increased set of criteria, and the results for ensembles constructed for the smaller criterion and then extended to a larger one. The later results are denoted with a prefix "E-" and followed by the suffix "-X-Y->Z," where X denotes the combination method that was used, Y denotes the size of the original ensemble, and Z denotes the size of the ensemble to which it was extended. Also, two stopping conditions were tested, with 1 000 and 10 000 ensembles evaluations.

Unfortunately, the results show that, although such a strategy of extending ensembles is possible, it does not actually lead to better results, or to any improvement in the time required to obtain the results, in comparison to constructing ensembles from scratch. Both methods usually achieved quite similar results, and usually there were no significant differences between them. However, there have also been several cases, especially when using the smaller termination criterion, where the extended ensembles performed significantly worse than those constructed from scratch. Also, we can observe that when considering problems with Mus criterion, ensembles using the vote combination method had an inferior performance. However, it is interesting to note that when this criterion was not included, the extended ensembles that used the vote combination method did result in better results than those using the sum combination method. Thus, it seems that the vote combination method is more resilient to the structure of the ensembles that are used as the bases for extension.

In the end, we can conclude that using a preconstructed set of ensembles is not that much helpful to the algorithm. However, the ensembles were extended only in a very basic way, and it could be possible that by using more sophisticated extension schemes the results could be improved. Thus, this constitutes a possible future research direction that can be investigated further.

Table 17: Results obtained by extending ensembles with additional rules to optimise problems with more objectives

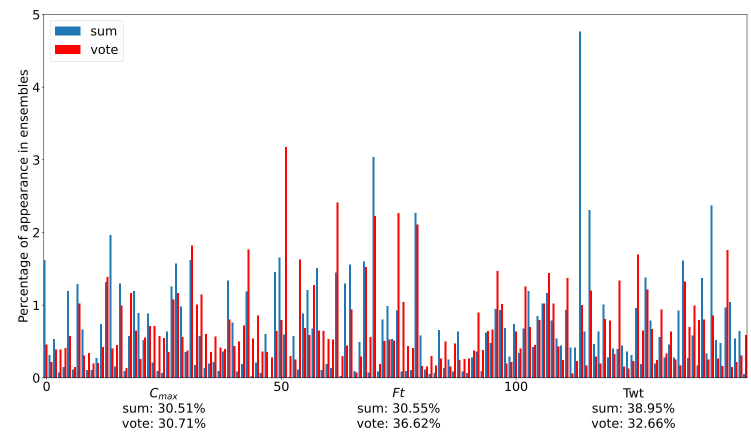
	Original criteria set	C_{max}, Ft, Mus, Twt	C_{max}, Ft, Nwt, Twt	C_{max}, Ft, Twt	C_{max}, Ft, Twt	C_{max}, Ft, Twt	C_{max}, Nwt, Twt	C_{max}, Nwt, Twt	C_{max}, Twt	C_{max}, Twt
	Increased criteria set	C_{max}, Ft, Mus, Nwt, Twt	C_{max}, Ft, Mus, Nwt, Twt	C_{max}, Ft, Mus, Nwt, Twt	C_{max}, Ft, Nwt, Twt	C_{max}, Ft, Mus, Twt	C_{max}, Ft, Nwt, Twt	C_{max}, Nwt, Twt	C_{max}, Ft, Twt	C_{max}, Ft, Nwt, Twt
10000 evaluations	SEC-S-3	0.629	0.639	0.639	0.723	0.742	0.723	0.639	0.846	0.740
	SEC-S-5	0.635	**0.645**	**0.645**	0.708	0.743	0.708	**0.645**	0.849	0.725
	SEC-S-7	0.632	0.642	0.642	0.699	**0.744**	0.699	0.642	0.843	0.716
	SEC-V-3	0.550	0.563	0.562	0.757	0.667	0.757	0.562	0.851	0.779
	SEC-V-5	0.565	0.578	0.577	0.767	0.674	0.767	0.577	0.853	0.788
	SEC-V-7	0.562	0.575	0.574	0.768	0.676	0.768	0.574	**0.854**	**0.790**
	E-SEC-S-3->5	0.652	0.642	0.630	0.715	0.739	0.712	0.633	0.844	0.701
	E-SEC-S-3->7	**0.653**	0.644	0.635	0.704	0.741	0.704	0.634	0.841	0.707
	E-SEC-S-5->7	**0.653**	0.633	0.632	0.702	0.741	0.713	0.629	0.842	0.694
	E-SEC-V-3->5	0.582	0.576	0.544	**0.773**	0.666	**0.771**	0.557	0.836	0.785
	E-SEC-V-3->7	0.572	0.579	0.554	0.769	0.671	0.759	0.562	0.846	0.779
	E-SEC-V-5->7	0.579	0.576	0.540	0.764	0.661	0.752	0.544	0.800	0.773
1000 evaluations	SEC-S-3	0.588	0.599	0.599	0.680	0.714	0.680	0.599	0.831	0.695
	SEC-S-5	0.594	0.605	0.605	0.668	0.715	0.668	0.605	0.833	0.683
	SEC-S-7	0.589	0.600	0.600	0.675	0.714	0.675	0.600	0.830	0.690
	SEC-V-3	0.500	0.514	0.513	0.723	0.624	0.723	0.513	0.832	0.742
	SEC-V-5	0.496	0.509	0.509	0.731	0.634	0.731	0.509	0.842	0.751
	SEC-V-7	0.503	0.517	0.517	0.736	0.623	0.736	0.517	0.843	0.756
	E-SEC-S-3->5	0.576	0.590	0.481	0.674	0.658	0.695	0.518	0.663	0.508
	E-SEC-S-3->7	0.576	0.590	0.481	0.674	0.658	0.695	0.518	0.663	0.508
	E-SEC-S-5->7	0.587	0.608	0.494	0.658	0.675	0.684	0.508	0.765	0.587
	E-SEC-V-3->5	0.442	0.506	0.369	0.738	0.558	0.720	0.378	0.739	0.640
	E-SEC-V-3->7	0.442	0.506	0.369	0.738	0.558	0.720	0.378	0.739	0.640
	E-SEC-V-5->7	0.475	0.516	0.433	0.748	0.582	0.713	0.380	0.723	0.655

7.5. Rule frequency in ensembles

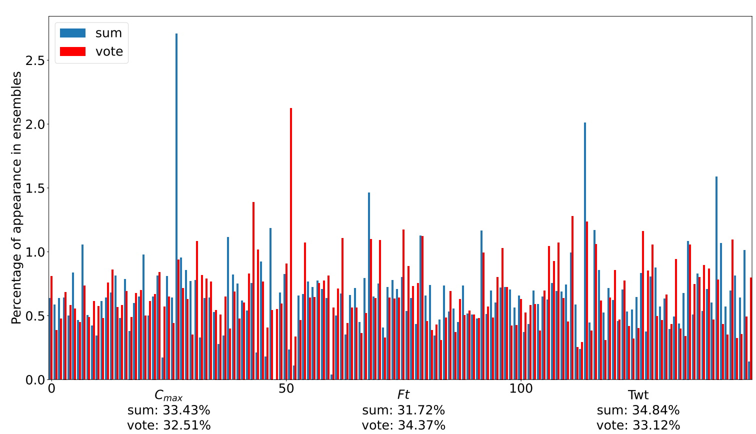
Another interesting thing to analyse is the composition of the ensembles in the Pareto fronts constructed by SEC. For this purpose, all Pareto fronts for three selected problems were aggregated and denoted in the form of bar plots. Furthermore, the plots are outlined for ensemble sizes 3 and 7 (since the dispersion for size 5 are not too different, this value was skipped), and both combination methods.

Figure 13 outlines the frequencies of the rules contained in the ensemble for the $R|r_j|C_{max}$, Ft, Twt problem. The x axis represents the ID of the rule that was used, grouped by criterion (denoted beneath each group) and sorted by the performance they obtained on the criterion they were optimised. This means that rules 1-50 were optimised for the C_{max} criterion, with rule 1 being the best one, and rule 50 the worst. In addition, the percentages of rules from each group being used in ensembles with the sum and vote combination methods are also denoted beneath each group.

First of all, it can be seen that rules from each group appear roughly in the same extent in the ensembles. The differences are more apparent for individual rules, as there are rules that appear more commonly than others. Also, there are some rules which are rarely used to construct the ensembles. However, most of the rules can be used in ensembles, and as such this means that having more rules is beneficial for the method, as they provide more diversity for constructing ensembles. Furthermore, it can also be seen that there are differences in the selection of rules between the ensembles that use the sum and vote combination method, which suggests that certain rules might be more suitable for one kind of combination method. Regarding the ensemble size, we do not observe a large change in the distributions, although we can see that for the sum combination method some rules now appear more often when larger ensembles are used. On the other hand, the distribution for the vote construction method seems more stable, and thus it seems that the choice of rules used in the ensemble does not change that much depending on the size of the ensembles. This is expected as the vote combination method is more resilient to rules which perform completely different decisions than all other rules in the ensemble.



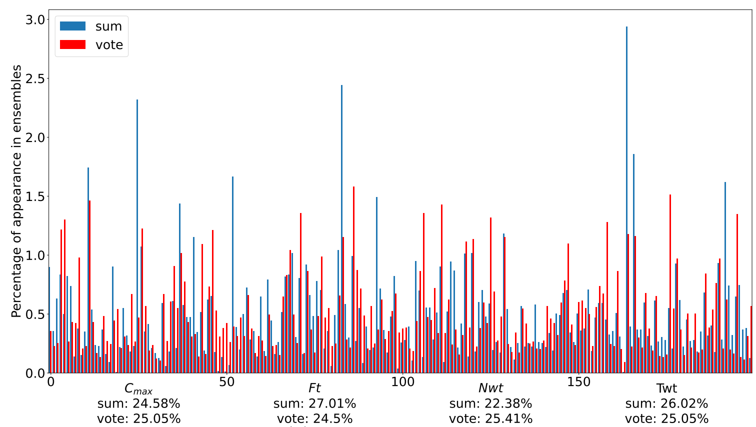
(a) Ensemble size of 3.



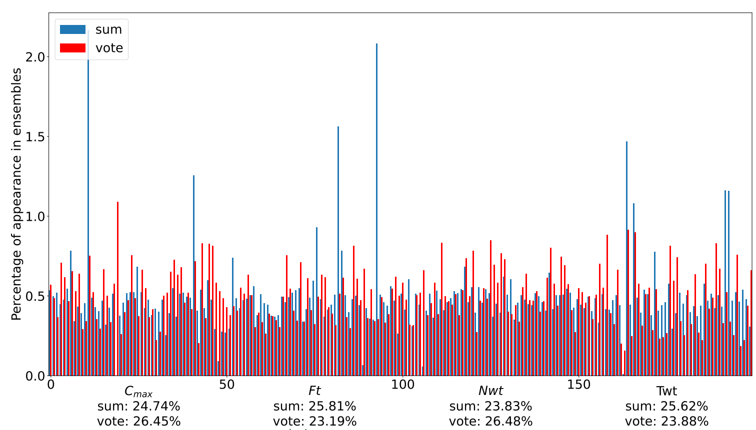
(b) Ensemble size of 7.

Figure 13: Frequency of rules in the constructed ensembles for the $R|r_j|C_{max}$, Ft, Twt problem.

Figure 14 shows the rule frequency in ensembles when considering the $R|r_j|C_{max}$, Ft, Twt problem. Again, it can be seen that rules evolved for each criterion participate in the construction of ensembles with a similar ratio. Again, most rules are used in the ensembles, which can best be seen for ensembles of size 7, where especially for the vote combination method the rules have a similar frequency of being used in the ensemble. However, there are certain rules which are rarely or even never used. For the sum combination method we can again observe that it does tend to prefer the selection of some rules in the ensemble, and that this slightly changes as the size of the ensemble is increased.



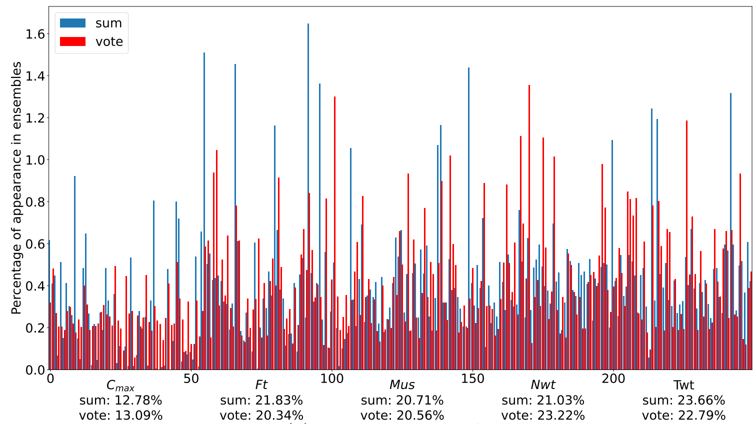
(a) Ensemble size of 3.



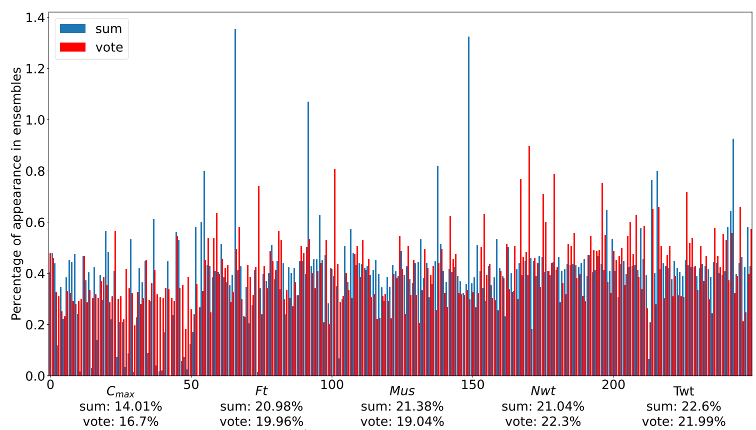
(b) Ensemble size of 7.

Figure 14: Frequency of rules in the constructed ensembles for the $R|r_j|C_{max}$, Ft, Twt problem.

Figure 15 shows the rule frequency in ensembles when considering the $R|r_j|C_{max}$, Ft, Mus, Nwt, Twt problem. Here, we can actually observe a slight difference in comparison with the previous figures. Namely, in this case, the rules evolved for the C_{max} criterion are used less frequently than other rules. This is an interesting behaviour that was also observed for some other problems as well. The reason for this could be due to the fact that rules evolved for optimising the C_{max} criterion that perform well usually did not perform well on the other criteria, which was seen from the correlation coefficients between this and other criteria obtained for the rules evolved for C_{max}. As rules for other criteria (like Ft) had even a positive correlation with others, maybe these rules were then more likely to get selected as it was easier to construct an ensemble that performs well across all criteria by including such rules. The only additional thing that can also be observed is that for ensembles of size 3, the vote combination method now also favours certain rules, but as the size increases to 7, the frequency of being included in the ensemble is more evenly distributed across all the rules.



(a) Ensemble size of 3.



(b) Ensemble size of 7.

Figure 15: Frequency of rules in the constructed ensembles for the $R|r_j|C_{max}$, Ft, Nwt, Twt, Mus problem.

Across all three problems it was seen that often the rules that are most commonly selected to build the ensembles are not necessarily the best rules evolved for the considered criterion. Instead, the ensembles usually consisted of rules that performed well across all the considered criteria, and had values similar to the median values denoted in Table 13 for the considered criteria. Therefore, it seems that it is not too important for the set of rules to include many rules with an outstanding performance for one criterion, but rather to have more rules that perform well across all criteria, as those can be better utilised when constructing ensembles.

7.6. Performance of individual rules

Table 18 outlines the performance of several selected rules evolved by NSGA-III and ensembles constructed by SEC. The values for the criteria are outlined only for those criteria for which they were evolved. Furthermore, for each problem, the number of rules and ensembles selected and denoted in the table is equal to the number of criteria considered in the problem. The rules and ensembles were selected in such a way that, for each criterion, a rule and ensemble that work well on that criterion were selected, with the addition that the rule and ensemble have a similar performance on it so that they can be assessed on how well they work on other criteria. The criterion for which the ensembles and rules were selected are denoted in boldface.

From the table, it is evident that neither the ensembles nor MO DRs always end up with the best results across all criteria. This can best be seen for the $R|r_j|C_{max}, Twt$ problem, in which the MO DR selected for optimising the Twt criterion performed better on C_{max} than the ensemble. However, when selecting the rules and ensembles based on their performance on C_{max}, then the selected ensemble achieved a better performance for Twt, by around 50%. This seems to suggest that when focusing on optimising C_{max} MO DRs have a difficult time to perform well on Twt as well, which does not seem to be the case with ensembles. This behaviour can also be noticed for all the other problems as well. More generally, it seems that MO DRs that perform better for the C_{max} criterion tend to perform poorly on all other criteria.

The downside of the ensembles is again connected to the problems that include the Mus criterion. Particularly, the ensembles that perform well for the Mus criterion achieve a poor performance on all the other criteria. This again serves to show that ensembles have trouble with criteria that are highly conflicting with the others. However, when selecting the rules for other criteria, the difference is not that significant, which seems to suggest that SEC simply has difficulties in finding good ensembles at the extreme points for the Mus criterion.

8. Findings and discussion

The obtained results and analyses performed in the previous sections have shown several interesting things about the proposed methodology and MO optimisation that will be summarised and shortly discussed in this section.

First of all, the experimental results indicate that the proposed methodology of using DRs evolved for optimising a SO to construct ensembles for optimising MO problems is plausible. Out of the 8 considered MO problems with different sizes and criteria compositions, in 5 the applied SEC method produced Pareto fronts that are significantly better than those obtained by MO DRs evolved either by NSGA-II or NSGA-III. In the three remaining problems, the Pareto fronts of ensembles were, in the best case, able to match the performance of Pareto fronts of DRs, meaning that there was no significant difference between the results obtained by the two methods. A deeper analysis demonstrated that this is due to the fact that these problems included a criterion that is highly conflicting with all the other criteria, namely the Mus criterion. However, since most standard scheduling criteria (C_{max}, Ft, Nwt, Twt, and others) are not highly conflicting with each other, this does not represent a significant issue for the proposed method. Particularly, since when applied to problems that include only criteria that are not highly conflicting, the ensembles significantly outperform the evolved MO DRs.

The experiments with the ensemble construction method have shown that SEC is already powerful enough to construct high quality ensembles, and that using either E-NSGA-II or E-NSGA-III for that purpose does not improve the results. As such, it makes sense to use SEC as it is simpler and less computationally expensive for constructing ensembles than using MO algorithms for that purpose. Furthermore, the results also show that SEC could already match the performance of MO DRs using a smaller number of rules evolved for each

Table 18: Performance of selected rules and ensembles across criteria on which they were evolved for

		C_{max}	Ft	Mus	Nwt	Twt
C_{max}	NSGA-III	**37.98**				21.64
	SEC	**37.97**				14.15
Twt	NSGA-III	38.38				**12.73**
	SEC	38.83				**12.74**
C_{max}	NSGA-III	**37.97**		0.120		
	SEC	**37.98**		0.123		
Mus	NSGA-III	57.77		**0.476**		
	SEC	67.43		**0.472**		
C_{max}	NSGA-III	**37.96**	158.9			19.26
	SEC	**37.99**	156.6			14.43
Ft	NSGA-III	38.63	**153.5**			15.63
	SEC	38.24	**153.5**			16.72
Twt	NSGA-III	38.83	173.7			**12.87**
	SEC	38.52	178.5			**12.87**
C_{max}	NSGA-III	**37.95**	173.1		8.427	27.44
	SEC	**37.98**	158.5		6.453	14.76
Ft	NSGA-III	38.40	**153.5**		7.097	16.93
	SEC	38.29	**153.4**		6.888	16.11
Nwt	NSGA-III	39.35	191.0		**6.246**	12.98
	SEC	39.34	173.5		**6.247**	13.14
Twt	NSGA-III	38.53	174.2		6.37	**12.72**
	SEC	38.75	181.8		6.503	**12.72**
C_{max}	NSGA-III	**37.97**	183.5	0.129	8.572	25.86
	SEC	**37.97**	176.3	0.127	7.533	18.01
Ft	NSGA-III	38.45	**153.5**	0.131	7.147	17.10
	SEC	38.61	**153.5**	0.138	7.109	16.79
Mus	NSGA-III	61.07	1197	**0.049**	37.71	407.9
	SEC	74.98	1519	**0.049**	43.14	664.8
Nwt	NSGA-III	38.37	178.8	0.134	**6.384**	13.05
	SEC	38.98	191.9	0.135	**6.387**	13.25
Twt	NSGA-III	38.35	178.7	0.135	6.349	**13.03**
	SEC	39.56	213.2	0.143	6.679	**13.05**

criterion, but also in a quite short amount of time. However, the best results were achieved when using larger sets of SO rules and given more ensemble evaluations.

On the other hand, regarding the ensemble parameters, we saw that the combination method had a larger effect on the quality of the results, than the ensemble size. The vote combination method was usually better for problems without the Mus criterion, whereas the sum combination method performed better on problems that included this criterion. As such, we can conclude that the sum combination method is more stable across different MO problems. There are probably two reasons for such a behaviour. First, the sum combination method can, in theory, produce more distinct ensembles due to the way in which the ensembles are interpreted, since each rule added to the ensemble could change the decisions of it. In contrast, when using a voting scheme, as long as the majority of rules perform the same decisions, adding a new rule will not affect the decisions of the ensemble. Second, due to the same property, it is more difficult to obtain ensembles that will provide a trade-off between Mus and the other criteria. The reason for this is that the ensemble will either contain more rules that are optimised for Mus or the other criteria, and therefore the entire ensemble will also be biased to perform better for the criteria for which it contains the majority of rules. On the other hand, the sum combination method can provide a better trade-off between the criteria as it uses the priority values directly, and can thus find ensembles in which the rules complement each other. Regarding the ensemble size, it rarely had a significant effect on the results, and in many cases the smallest ensembles of size three already performed well enough.

A very interesting behaviour, both for MO rules and ensembles, was observed when performing additional analyses. Namely, it was shown that Pareto fronts obtained for larger problems that included more criteria, could be also utilised efficiently for smaller problems with a subset of criteria, without any loss in the quality of the Pareto front. This is an interesting observation, which unfortunately can only be applied for Pareto fronts of heuristics. Nevertheless, it represents a valuable finding since it means that it is not required to obtain Pareto fronts of DRs for all different MO problems, but rather only for a few selected ones, and that these Pareto fronts also obtain good results when used on problems that include a smaller number of criteria.

Thus, the main findings of this paper can be summarised as follows:

- DRs evolved for optimising SO problems can be effectively combined into ensembles that are suitable for optimising MO problems.
- The ensembles can be constructed using the SEC method that randomly selects rules that should form the ensemble, and a more complex metaheuristic is unnecessary.
- For most problems, ensembles achieved a significantly better performance than MO DRs, and for the problems in which this did not happen, ensembles were still able to match the performance of DRs.
- Pareto fronts of both MO DRs and ensembles show a neat generalisation ability, by which it is possible to reutilise Pareto fronts obtained for larger problems on smaller problems.

9. Conclusions and Future Work

In this study, a novel methodology was proposed for creating ensembles of DRs for MO problems by using DRs evolved for individual criteria. Unlike the standard approach in which various MO algorithms are used to generate new DRs, this method combines rules evolved in SO optimisation to obtain ensembles which can efficiently optimise multiple criteria. As such, the goal of this method is to reutilise existing high quality rules for MO problems.

The experiments showed that the ensembles constructed by the proposed methodology are capable of achieving better or equal results in comparison to DRs evolved especially for MO problems. These results show the effectiveness of such an alternative approach to solving MO problems. This shows that using ensembles in the domain of MO optimisation is reasonable, especially as these two areas were seldom considered in combination. From the additional analyses, a quite interesting behaviour was also observed, namely that Pareto fronts of DRs or ensembles evolved for larger number of criteria can be reutilised for smaller criteria combinations without any performance penalty. This makes it possible to evolve rules or ensembles for larger problems and then reuse them for smaller ones without having to evolve new rules or construct new ensembles. Although this observation is something that can be applied only on Pareto fronts of heuristics, it nevertheless represents a quite interesting feature when applying hyper-heuristic methods in the context of MO problems.

The obtained results and analyses open up several directions in which this research could be extended in the future. First of all, it is important to address the weaker performance of the proposed methodology when considering highly conflicting criteria. Our hypothesis is that this happens due to the too small diversity of the initial rules in such problems. However, more investigation is required to identify more precisely the cause and to propose remedies for it. Secondly, the SEC method in its basis constructs the ensembles completely in a random way. Although this approach has proven to be powerful enough, it would still be interesting to investigate whether a better method for constructing the ensembles could result in improved results. Naturally, the main challenge here lies in how to determine which rules should be included in the ensemble considering MO problems. Finally, we would also like to employ the proposed methodology also on other environments in which hyper-heuristics can be used, in order to determine how general the proposed approach is across various domains.

Acknowledgement

This work has been supported in part by Croatian Science Foundation under the project IP-2019-04-4333 and by the Spanish State Agency for Research (AEI) under research project PID2019-106263RB-I00. Carlos A. Coello Coello gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (Investigación en Fronteras de la Ciencia 2016). He was also partially supported by the Basque Government through the BERC 2022-2025 program and by Spanish Ministry of Economy and Competitiveness MINECO: BCAM Severo Ochoa excellence accreditation SEV-2017-0718.

References

[1] M. L. Pinedo, Scheduling, Springer US, 2012. doi:10.1007/978-1-4614-2361-4.

[2] L. Wu, S. Wang, Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks, International Journal of Production Economics 201 (2018) 26–40. doi:https://10.1016/j.ijpe.2018.04.013.

[3] R. Gedik, D. Kalathia, G. Egilmez, E. Kirac, A constraint programming approach for solving unrelated parallel machine scheduling problem, Computers & Industrial Engineering 121 (2018) 139–149. doi:https://10.1016/j.cie.2018.05.014.

[4] L. Yu, H. M. Shih, M. Pfund, W. M. Carlyle, J. W. Fowler, IIE Transactions 34 (11) (2002) 921–931. doi:10.1023/a:1016185412209.

[5] S. N. Makhadmeh, A. T. Khader, M. A. Al-Betar, S. Naim, A. K. Abasi, Z. A. A. Alyasseri, Optimization methods for power scheduling problems in smart home: Survey, Renewable and Sustainable Energy Reviews 115 (2019) 109362. doi:doi.org/10.1016/j.rser.2019.109362.

[6] S. N. Makhadmeh, A. T. Khader, M. A. Al-Betar, S. Naim, A. K. Abasi, Z. A. A. Alyasseri, A novel hybrid grey wolf optimizer with min-conflict algorithm for power scheduling problem in a smart home, Swarm and Evolutionary Computation 60 (2021) 100793. doi:doi.org/10.1016/j.swevo.2020.100793.

[7] E. Hart, P. Ross, D. Corne, Evolutionary Scheduling: A Review, Genetic Programming and Evolvable Machines 6 (2) (2005) 191–220. doi:10.1007/s10710-005-7580-7.

[8] I. Vlašić, M. Đurasević, D. Jakobović, Improving genetic algorithm performance by population initialisation with dispatching rules, Computers & Industrial Engineering 137 (2019) 106030. doi:https://10.1016/j.cie.2019.106030.

[9] J.-P. Arnaout, G. Rabadi, R. Musa, 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 (6) (2009) 693–701. doi:10.1007/s10845-009-0246-1.

[10] J.-P. Arnaout, R. Musa, G. Rabadi, A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines—part II: enhancements and experimentations, Journal of Intelligent Manufacturing 25 (1) (2012) 43–53. doi:10.1007/s10845-012-0672-3.

[11] G. Bektur, T. Sarac, A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server, Computers & Operations Research 103 (2019) 46–63. doi:https://10.1016/j.cor.2018.10.010.

[12] S. Wang, Y. Mei, J. Park, M. Zhang, Evolving ensembles of routing policies using genetic programming for uncertain capacitated arc routing problem, in: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 1628–1635. doi:10.1109/SSCI44817.2019.9002749.

[13] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, Q. Zhang, A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times, IEEE Transactions on Cybernetics 51 (3) (2021) 1430–1442. doi:10.1109/TCYB.2019.2939219.

[14] L. Fanjul-Peyro, R. Ruiz, Iterated greedy local search methods for unrelated parallel machine scheduling, European Journal of Operational Research 207 (1) (2010) 55–69. doi:https://10.1016/j.ejor.2010.03.030.

[15] L. Ulaga, M. Đurasević, D. Jakobović, Local search based methods for scheduling in the unrelated parallel machines environment, Expert Systems with Applications 199 (2022) 116909. doi:https://10.1016/j.eswa.2022.116909.

[16] F. Zhao, Z. Xu, L. Wang, N. Zhu, T. Xu, Jonrinaldi, A population-based iterated greedy algorithm for distributed assembly no-wait flow-shop scheduling problem, IEEE Transactions on Industrial Informatics (2022) 1–12doi:10.1109/TII.2022.3192881.

[17] M. Đurasević, D. Jakobović, A survey of dispatching rules for the dynamic unrelated machines environment, Expert Systems with Applications 113 (2018) 555–569. doi:https://10.1016/j.eswa.2018.06.053.

[18] J. Branke, S. Nguyen, C. W. Pickardt, M. Zhang, Automated design of production scheduling heuristics: A review, IEEE Transactions on Evolutionary Computation 20 (1) (2016) 110–124. doi:10.1109/TEVC.2015.2429314.

[19] S. Nguyen, Y. Mei, M. Zhang, Genetic programming for production scheduling: a survey with a unified framework, Complex & Intelligent Systems 3 (1) (2017) 41–66. doi:10.1007/s40747-017-0036-x.

[20] R. Poli, W. B. Langdon, N. F. McPhee, A Field Guide to Genetic Programming, Lulu Enterprises, UK Ltd, 2008.

[21] M. Đurasević, D. Jakobović, K. Knežević, Adaptive scheduling on unrelated machines with genetic programming, Applied Soft Computing 48 (2016) 419–430. doi:https://10.1016/j.asoc.2016.07.025.

[22] S. Nguyen, M. Zhang, M. Johnston, K. C. Tan, Dynamic multi-objective job shop scheduling: A genetic programming approach, in: Studies in

Computational Intelligence, Springer Berlin Heidelberg, 2013, pp. 251–282. doi:10.1007/978-3-642-39304-4_10.

[23] D. Jakobović, L. Budin, Dynamic scheduling with genetic programming, in: P. Collet, M. Tomassini, M. Ebner, S. Gustafson, A. Ekárt (Eds.), Genetic Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 73–84.

[24] F. J. Gil-Gala, R. Varela, 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, pp. 223–233. doi:10.1007/978-3-030-19651-6_22.

[25] M. Đumić, D. Šišejković, R. Čorić, D. Jakobović, Evolving priority rules for resource constrained project scheduling problem with genetic programming, Future Generation Computer Systems 86 (2018) 211–221. doi:https://10.1016/j.future.2018.04.029.

[26] Y. Cui, Z. Geng, Q. Zhu, Y. Han, Review: Multi-objective optimization methods and application in energy saving, Energy 125 (2017) 681–704. doi:https://doi.org/10.1016/j.energy.2017.02.174.
URL https://www.sciencedirect.com/science/article/pii/S0360544217303584

[27] S. Y. Ivanov, A. K. Ray, Application of multi-objective optimization in the design and operation of industrial catalytic reactors and processes, Physical Sciences Reviews 1 (3) (2016) 20150017 [cited 2023-03-14]. doi:doi:10.1515/psr-2015-0017.
URL https://doi.org/10.1515/psr-2015-0017

[28] J. S. Neufeld, S. Schulz, U. Buscher, A systematic review of multi-objective hybrid flow shop scheduling, European Journal of Operational Researchdoi:https://doi.org/10.1016/j.ejor.2022.08.009.

URL https://www.sciencedirect.com/science/article/pii/S037722172200652X

[29] R. H. Stewart, T. S. Palmer, B. DuPont, A survey of multi-objective optimization methods and their applications for nuclear scientists and engineers, Progress in Nuclear Energy 138 (2021) 103830. doi:https://doi.org/10.1016/j.pnucene.2021.103830.
URL https://www.sciencedirect.com/science/article/pii/S0149197021001931

[30] R. Marler, J. Arora, Survey of multi-objective optimization methods for engineering, Structural and Multidisciplinary Optimization 26 (6) (2004) 369–395. doi:10.1007/s00158-003-0368-6.
URL https://doi.org/10.1007/s00158-003-0368-6

[31] N. Gunantara, A review of multi-objective optimization: Methods and its applications, Cogent Engineering 5 (1) (2018) 1502242. doi:10.1080/23311916.2018.1502242.

[32] I. Giagkiozis, P. Fleming, Methods for multi-objective optimization: An analysis, Information Sciences 293 (2015) 338–350. doi:https://doi.org/10.1016/j.ins.2014.08.071.
URL https://www.sciencedirect.com/science/article/pii/S0020025514009074

[33] s. Sharma, V. Chahar, A comprehensive review on multi-objective optimization techniques: Past, present and future, Archives of Computational Methods in Engineering 29 (2022) 3. doi:10.1007/s11831-022-09778-9.

[34] M. Đurasević, D. Jakobović, Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey, Artificial Intelligence Reviewdoi:10.1007/s10462-022-10247-9.
URL https://doi.org/10.1007/s10462-022-10247-9

[35] M. Đurasević, D. Jakobović, Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment, Genetic Programming and Evolvable Machines 19 (1-2) (2017) 9–51. doi:10.1007/s10710-017-9310-3.

[36] J. Park, S. Nguyen, M. Zhang, M. Johnston, Evolving ensembles of dispatching rules using genetic programming for job shop scheduling, in: P. Machado, M. I. Heywood, J. McDermott, M. Castelli, P. García-Sánchez, P. Burelli, S. Risi, K. Sim (Eds.), Genetic Programming, Springer International Publishing, Cham, 2015, pp. 92–104.

[37] M. Đurasević, D. Jakobović, Creating dispatching rules by simple ensemble combination, Journal of Heuristics 25 (6) (2019) 959–1013. doi:10.1007/s10732-019-09416-x.

[38] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, J. R. Woodward, Exploring Hyper-heuristic Methodologies with Genetic Programming, Computational Intelligence 1 (2009) 177–201. doi:10.1007/978-3-642-01799-5_6.

[39] J. Jacobsen-Grocott, Y. Mei, G. Chen, M. Zhang, Evolving heuristics for dynamic vehicle routing with time windows using genetic programming, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1948–1955. doi:10.1109/CEC.2017.7969539.

[40] G. Duflo, E. Kieffer, M. R. Brust, G. Danoy, P. Bouvry, A gp hyper-heuristic approach for generating tsp heuristics, in: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2019, pp. 521–529. doi:10.1109/IPDPSW.2019.00094.

[41] Y. Liu, Y. Mei, M. Zhang, Z. Zhang, A Predictive-Reactive Approach with Genetic Programming and Cooperative Coevolution for the Uncertain Capacitated Arc Routing Problem, Evolutionary Computation 28 (2) (2020) 289–316. arXiv:https://direct.mit.edu/evco/article-pdf/28/2/289/1858868/evco_a_00256.pdf, doi:10.1162/evco_a_00256.

[42] S. Nguyen, M. Zhang, M. Johnston, K. C. Tan, A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem, IEEE Transactions on Evolutionary Computation 17 (5) (2013) 621–639. doi:10.1109/TEVC.2012.2227326.

[43] S. Chand, Q. Huynh, H. Singh, T. Ray, M. Wagner, On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems, Information Sciences 432 (2018) 146–163. doi:https://10.1016/j.ins.2017.12.013.

[44] F. Zhang, Y. Mei, S. Nguyen, M. Zhang, K. C. Tan, Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling, IEEE Transactions on Evolutionary Computation 25 (4) (2021) 651–665. doi:10.1109/TEVC.2021.3065707.

[45] F. Zhang, Y. Mei, S. Nguyen, M. Zhang, Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling, IEEE Transactions on Cybernetics 51 (4) (2021) 1797–1811. doi:10.1109/TCYB.2020.3024849.

[46] F. J. Gil-Gala, M. R. Sierra, C. Mencía, R. Varela, Genetic programming with local search to evolve priority rules for scheduling jobs on a machine with time-varying capacity, Swarm and Evolutionary Computation 66 (2021) 100944. doi:https://10.1016/j.swevo.2021.100944.

[47] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, M. Zhang, Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling, IEEE Transactions on Cybernetics (2021) 1–14doi:10.1109/TCYB.2021.3065340.

[48] K. Jaklinović, M. Đurasević, D. Jakobović, Designing dispatching rules with genetic programming for the unrelated machines environment with constraints, Expert Systems with Applications 172 (2021) 114548. doi:https://10.1016/j.eswa.2020.114548.

[49] M. Đurasević, D. Jakobović, Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics, Journal of Computational Science 61 (2022) 101649. doi:https://10.1016/j.jocs.2022.101649.

[50] S. Nguyen, M. Zhang, K. C. Tan, Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 2781–2788. doi:10.1109/CEC.2015.7257234.

[51] A. Masood, Y. Mei, G. Chen, M. Zhang, Many-objective genetic programming for job-shop scheduling, in: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 209–216. doi:10.1109/CEC.2016.7743797.

[52] F. Zhang, Y. Mei, M. Zhang, 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, pp. 1366–1373. doi:10.1109/CEC.2019.8790112.

[53] F. Zhang, S. Nguyen, Y. Mei, M. Zhang, Learning scheduling heuristics for multi-objective dynamic flexible job shop scheduling, in: Genetic Programming for Production Scheduling, Springer Singapore, 2021, pp. 235–245. doi:10.1007/978-981-16-4859-5_12.

[54] A. Masood, G. Chen, Y. Mei, H. Al-Sahaf, M. Zhang, Genetic programming with pareto local search for many-objective job shop scheduling, in: AI 2019: Advances in Artificial Intelligence, Springer International Publishing, 2019, pp. 536–548. doi:10.1007/978-3-030-35288-2_43.

[55] A. Masood, G. Chen, Y. Mei, H. Al-Sahaf, M. Zhang, A fitness-based selection method for pareto local search for many-objective job shop scheduling, in: 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1–8. doi:10.1109/CEC48606.2020.9185881.

[56] J. Park, Y. Mei, S. Nguyen, G. Chen, M. Johnston, M. Zhang, Genetic programming based hyper-heuristics for dynamic job shop scheduling: Cooperative coevolutionary approaches, in: Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 115–132. `doi:10.1007/978-3-319-30668-1_8`.

[57] E. Hart, K. Sim, A hyper-heuristic ensemble method for static job-shop scheduling, Evolutionary Computation 24 (4) (2016) 609–635. `doi:10.1162/EVCO_a_00183`.

[58] J. Park, Y. Mei, S. Nguyen, G. Chen, M. Zhang, An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling, Applied Soft Computing 63. `doi:10.1016/j.asoc.2017.11.020`.

[59] M. Đurasević, D. Jakobović, Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment, Genetic Programming and Evolvable Machines 19 (1-2) (2017) 53–92. `doi:10.1007/s10710-017-9302-3`.

[60] M. Đumić, D. Jakobović, Ensembles of priority rules for resource constrained project scheduling problem, Applied Soft Computing 110 (2021) 107606. `doi:https://10.1016/j.asoc.2021.107606`.

[61] F. J. Gil-Gala, M. R. Sierra, C. Mencía, R. Varela, Combining hyper-heuristics to evolve ensembles of priority rules for on-line scheduling, Natural Computing`doi:10.1007/s11047-020-09793-4`.

[62] F. J. Gil-Gala, C. Mencía, M. R. Sierra, R. Varela, Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms, Integrated Computer-Aided Engineering 28 (1) (2020) 65–80. `doi:10.3233/ICA-200634`.

[63] M. Đurasević, L. Planinić, F. J. Gil-Gala, D. Jakobović, Novel ensemble collaboration method for dynamic scheduling problems, in: Proceedings of

the Genetic and Evolutionary Computation Conference, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 893–901. doi:10.1145/3512290.3528807.

[64] M. Đurasević, L. Planinić, F. J. Gil-Gala, D. Jakobović, Constructing ensembles of dispatching rules for multi-objective problems, in: J. M. Ferrández Vicente, J. R. Álvarez-Sánchez, F. de la Paz López, H. Adeli (Eds.), Bio-inspired Systems and Applications: from Robotics to Ambient Intelligence, Springer International Publishing, Cham, 2022, pp. 119–129.

[65] J. Branke, T. Hildebrandt, B. Scholz-Reiter, Hyper-heuristic Evolution of Dispatching Rules: A Comparison of Rule Representations, Evolutionary Computation 23 (2) (2015) 249–277. arXiv:https://direct.mit.edu/evco/article-pdf/23/2/249/1514450/evco_a_00131.pdf, doi:10.1162/EVCO_a_00131.
URL https://doi.org/10.1162/EVCO_a_00131

[66] L. Planinić, H. Backović, M. Đurasević, D. Jakobović, A comparative study of dispatching rule representations in evolutionary algorithms for the dynamic unrelated machines environment, IEEE Access 10 (2022) 22886–22901. doi:10.1109/ACCESS.2022.3151346.

[67] F. J. Gil-Gala, M. Đurasević, M. R. Sierra, R. Varela, Building heuristics and ensembles for the travel salesman problem, in: J. M. Ferrández Vicente, J. R. Álvarez-Sánchez, F. de la Paz López, H. Adeli (Eds.), Bio-inspired Systems and Applications: from Robotics to Ambient Intelligence, Springer International Publishing, Cham, 2022, pp. 130–139.