Collaboration methods for ensembles of dispatching rules for the dynamic unrelated machines environment

Marko Đurasević[a,*], Francisco Javier Gil Gala[b], Lucija Planinić[c], Domagoj Jakobović[a]

[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] *Google, Zürich, Switzerland*

Abstract

Dynamic scheduling represents an important combinatorial optimisation problem which often appears in the real world. The difficulty in solving these problems arises from their dynamic nature, which limits the applicability of improvement based metaheuristics. Dynamic problems are usually solved using dispatching rules (DRs), which iteratively construct the schedule. Recently such heuristics have been constructed using various hyperheuristic methods, most notably genetic programming. Although automatically designed DRs achieve good performance, it is still very difficult to design a *single* DR that would perform a good decision at every decision point. As a remedy, DRs were combined into ensembles to improve their performance. For that purpose it is required to define how ensembles are constructed and how DRs in the ensemble collaborate. This paper proposes a novel *ensemble collaboration* method based on a similar method applied for static scheduling problems and adapts it for dynamic problems. The goal is to obtain a collaboration method that produces better results than standard collaboration methods. Additionally, the paper investigates the application of novel *ensemble construction* methods for dynamic scheduling. The proposed methods are validated on dynamic unrelated machines scheduling

*Fully documented templates are available in the elsarticle package on CTAN.

*Corresponding author
Email address: support@elsevier.com (Domagoj Jakobović)
URL: www.elsevier.com (Marko Đurasević)

problem and compared with existing ensemble construction and collaboration methods. The obtained results demonstrate that the proposed collaboration method performs better than standard ones. Further analyses provide additional insights into the proposed methods and outline several potential research directions in the area of hyper-heuristic ensemble construction.
Keywords: genetic programming, unrelated machines environment, scheduling, dispatching rules, ensembles

1. Introduction

Scheduling problems represent an important combinatorial optimisation problem in which certain jobs have to be allocated to a limited number of available resources to optimise some user-defined criteria (Pinedo, 2012). Such problems are heavily investigated since they appear in many real-world situations like computer multiprocessor task scheduling (Wu & Wang, 2018), equipment scheduling (Gedik et al., 2018), and manufacturing (Yu et al., 2002). Solving such problems under dynamic conditions, in which not all information about the problems is known beforehand, represents a particularly difficult challenge. The reason for this is due to the dynamic nature of the problem (e.g. it is not known which jobs and at which time they appear in the system) traditional improvement based metaheuristic methods like genetic algorithms (Vlašić et al., 2019), simulated annealing (Anagnostopoulos & Rabadi, 2002), tabu search(Lee et al., 2013), variable neighourhood descent (Fanjul-Peyro & Ruiz, 2010), ant colony optimisation (Arnaout et al., 2012) and others (Hart et al., 2005) cannot be easily applied. Therefore, such problems are usually solved using simple constructive heuristics, called dispatching rules.

Dispatching rules (DRs) are simple heuristics that at each decision point decide which job to schedule on which machine (Đurasević & Jakobović, 2018). This means that they do not build the entire schedule up front, but rather each time at least one machine is available and some jobs are still not scheduled, the DR is invoked to perform the decision on the next job that should be scheduled

on a machine. In that way the DR can quickly react to any changes that happen in the system, since they perform immediate decisions and do not perform any decisions for the future of the system. However, DRs suffer from certain issues that different research directions tried to mitigate over the years.

One of the most important issues associated with DRs is the difficulty in manually designing new rules. Since many problem variants, criteria, and conditions exists, DRs would need to be designed for all such combinations when required. Naturally, this is almost impossible due to the sheer number of problem variants that exist, as well as the fact that the design of such rules is a time consuming process which requires expert knowledge. Therefore, different hyper-heuristic methods, most notably genetic programming (GP) (Poli et al., 2008), have been used over the years to automatically design new DRs (Branke et al., 2016; Nguyen et al., 2017). With such methods it is possible to efficiently design new DRs that outperform manually designed DRs.

However, the performance of automatically designed rules is still limited, and it is not possible to design a single rule which would perform well in all possible situations (Đurasević & Jakobović, 2022). Therefore, different ways of improving the performance of DRs were investigated. One of the most investigated research directions in that regard is how individual DRs could collaborate to achieve a better performance. This was most commonly achieved by combining individual DRs into ensembles that are collectively used to solve scheduling problems (Park et al., 2015). Previous studies demonstrated that by creating such ensembles the performance of individual DRs could be significantly improved (Park et al., 2015; Đurasević & Jakobović, 2017a).

One of the most important decisions in ensemble design is how the rules work together. So far, most studies have used a method that combines the decisions of the individual rules of the ensemble into a single decision, e.g. by using the summation and voting combination methods (Park et al., 2017). However, such methods have a limited overview of the problem, as they do not consider the possible impact of their decision on the future of the system. Therefore, in this study, we explore an alternative method where the DRs of the ensemble can

collaborate to obtain a better overview of how the current decision might affect future decisions. In this method, the rules are used independently to construct an approximation of the schedule by tentatively scheduling all currently available jobs. These approximations are used to determine which rule would perform the best scheduling decision at the current time, which is then selected and executed in the real schedule. With such a strategy, the ensemble obtains a notion on how each rule would perform in the longer term under the current system conditions and can select the most appropriate rule for the current decision point based on this. The initial results obtained for this collaboration method showed great potential compared to a single DR (Đurasević et al., 2022). In this study, we extend the investigation of this collaboration method in detail and compare it with the standard sum and vote collaboration methods used in most studies. The contributions of this study can be summarised as follows:

- A novel collaboration method for ensembles of DRs in dynamic scheduling environment
- Novel application of genetic algorithms to constructs ensembles of DRs
- Detailed analysis of various ensemble aspects for the evaluated collaboration methods

The rest of the paper is organised as follows. Section 2 outlines the related literature. Section 3 describes the considered scheduling problem and the way in which GP is used to automatically generate DRs. The ensemble collaboration and construction methods are described in Section 4. The setup of the experiments is described in Section 5. Section 6 outlines the obtained results. A deeper analysis of the obtained results is performed in Section 7. Finally, Section 8 gives a short conclusion and outlines potential directions for future work.

2. Literature review

In recent years, hyper-heuristics became an emerging area of research with application on a wide range of problems (Burke et al., 2018). Generally, hyper-heuristics can be classified as methods used either to select or generate heuristics for solving a certain optimisation problem. In the context of scheduling problems, hyper-heuristic methods based on heuristic selection have been rarely used. For example, Vázquez-Rodríguez & Petrovic (2009) applied a genetic algorithm to determine the sequence in which DRs should be applied to construct a schedule to a certain problem, whereas Ochoa et al. (2009) perform a fitness landscape analysis of the search space of selection hyper-heuristics, showing that they are "easy" to search. Nguyen et al. (2013) used GP to construct rules for selecting the appropriate DR given certain system parameters. However, this study demonstrated that such rules were inferior to DRs designed by generative hyper-heuristics. And although other studies still investigated methods used to select appropriate heuristics (Zahmani & Atmani, 2019; Đurasević & Jakobović, 2022), the main research direction was oriented towards the application of hyper-heuristics for generating new DRs.

Out of the many available hyper-heuristic methods used for heuristic generation, GP profiled itself as the most popular method used to generate new heuristics to solve various combinatorial problems (Burke et al., 2010, 2013). Although GP has been used to generate heuristics for various problems like the travelling salesman problem (Duflo et al., 2019), vehicle routing problem (Jacobsen-Grocott et al., 2017), capacitated arc routing problem (Liu et al., 2020; Ardeh et al., 2021), its most prominent use was in the automated design of dispatching rules for various scheduling problems (Branke et al., 2016; Nguyen et al., 2017).

Designing new DRs with GP was performed for various scheduling environments like the single machine (Gil-Gala et al., 2019), job shop (Nguyen et al., 2018), unrelated machines (Jaklinović et al., 2021), resource constrained project scheduling (Chand et al., 2018; Đumić et al., 2018), and flexible job shop (Zhang

et al., 2021c). In all of these, automatically designed DRs achieved a better performance than existing manually designed DRs. This motivated many different research directions to further investigate the application of GP for generating DRs and improving their performance. Some prominent research directions include investigation of different solution representations (Nguyen et al., 2013; Branke et al., 2015; Planinić et al., 2022), multi-objective optimisation (Nguyen et al., 2013; Đurasević & Jakobović, 2017b; Xu et al., 2021), surrogate models (Zhang et al., 2021e,b), feature selection (Zhang et al., 2021d, 2019), local search (Gil-Gala et al., 2021), multitask GP (Zhang et al., 2021f,a), and many others.

An important objective present in most research dealing with the automated design of DRs is to constantly increase the performance the generated rules. A popular way to achieve this is to apply ensemble learning, which has already demonstrated a great efficiency in improving the performance of various machine learning models (Ganaie et al., 2022). Therefore, several studies proposed new GP variants based on ensemble learning, such as BoostGP and BagGP (Iba, 1999; Paris et al., 2001), which were inspired by corresponding methods from the machine learning area. Over the years, ensemble learning methods in GP became increasingly popular and applied in different problem domains, such as in classification with unbalanced data (Bhowan et al., 2013), intrusion detection (Folino et al., 2005), pattern (Folino et al., 2008) and image classification (Fan et al., 2022). Recent years also saw the development of novel ensemble learning methods such as M3GP (Muñoz et al., 2015), Ensemble GP (Rodrigues et al., 2020), or diverse bagging and niching GP Wang et al. (2019b). This shows that ensemble learning in GP is still and actively researched topic, with the aim of continuously improving existing methods and results (Virgolin, 2021).

The first application of ensembles for scheduling problems was performed by Park et al. (2015). In this study the authors applied a cooperative coevolution GP algorithm that simultaneously evolved DRs in different subpopulations and combined them into ensembles to achieve a better performance. The DRs were evolved completely independently from each other, and only interacted when they needed to be evaluated, thus being combined with representative rules

from other subpopulations to form ensembles. The results demonstrated that the obtained ensembles were more robust than a single DR. Park et al. (2016) applied a multilevel GP method to evolve ensembles. The idea of this method is to perform the evolution in several levels, starting with the evolution of ensembles, and then switching to the evolution of individuals. However, the proposed method did not obtain any improvement over the ensembles evolved by Park et al. (2015), except from being able to evolve them in less time. (Hart & Sim, 2016) propose the application of NELLI-GP for the creation of ensembles. In this method DRs are evolved and combined into ensembles by trying to specialise DRs for different problem instances in the training set. The obtained results demonstrate that the proposed method performed better than the method from Park et al. (2015). Park et al. (2017) investigate different combination methods used to combine decisions of individual DRs in the ensemble. They apply four methods, summation, voting, weighted summation, and weighted voting. The experimental results show that the weighted variants of the methods achieved inferior results, as well as that the summation combination method achieved the best results on average.

In all previous studies the methods used to construct ensembles also evolved the DRs that constitute the ensemble. However, alternative methods were proposed which use already existing DRs to construct ensembles. Đurasević & Jakobović (2017a) propose a novel ensemble construction method denoted as SEC to create ensembles for the unrelated machines environment. This method uses a set of DRs generated by GP and combines them into an ensemble by simple random combination. The method was compared to other ensemble construction methods that also evolve DRs, such as BagGP, BoostGP, and cooperative coevolution. The obtained results demonstrated that SEC achieved a superior performance in comparison to other ensemble learning methods. The previously outlined methods were also applied by Đumić & Jakobović (2021) on the resource constrained project scheduling problem. The obtained results demonstrated that the SEC method again performed better than the other ensemble learning methods, even for this problem variant. Therefore, Đurasević

& Jakobović (2019) further investigated the SEC method with different ways of constructing ensembles, where it was demonstrated that efficient ensembles could be constructed with simple greedy heuristics or random search for ensembles.

A novel way in which DRs in ensembles can collaborate was proposed by Gil-Gala & Varela (2019). In these ensembles the DRs do not actually collaborate with each other, rather each DR is executed individually on the considered problem and then the best of the schedules obtained by any of the rules in the ensemble is retained. In addition, the authors use a GA to construct the ensembles. However, such ensembles can be used only for offline problems where the schedule can be constructed prior to the execution of the system, unlike the ensembles used previously in the literature. This type of ensembles were further investigated by (Gil-Gala et al., 2020b), where the authors compare ensembles constructed from the manually designed ATC rule and GP generated rules. The results showed that the ensembles constructed from the ATC rule could not compete with ensembles constructed from automatically generated DRs. (Gil-Gala et al., 2020a) applied different methods to construct ensembles, including an iterated greedy method, a local search method, and a memetic algorithm that achieved the best results. Aside from the application in scheduling, ensembles were also successfully applied on the capacitated arc routing problem as well (Wang et al., 2019b,a).

Based on the idea of the previously described ensemble type, a novel ensemble collaboration method was proposed for the dynamic scheduling environment (Đurasević et al., 2022). In this collaboration method, at each decision point all DRs in the ensemble are used independently to simulate the scheduling of all currently available jobs in the system. After that simulation, the rule that achieved the best value of the optimised criterion is executed to perform the real scheduling decision. In that way, by considering only the currently available jobs, such a method is applicable in dynamic conditions. The ensembles were constructed by random sampling with the SEC method, and the obtained results demonstrated that such ensembles were superior to individual rules. Be-

cause of the potential that this collaboration method has demonstrated, in this study we extend the analysis by using alternative ensemble construction methods and providing a detailed comparison with classical ensemble collaboration methods.

3. Background

3.1. Unrelated machines scheduling

In this study the parallel unrelated machines environment is investigated. This scheduling problem variant appears in many real-world situations like manufacturing, cloud environments and others (Pinedo, 2012). In this environment, a set of n jobs must be scheduled on one of the m available machines. In the considered problem several system properties are available, including:

- p_{ij} - processing time of job j on machine i,
- r_j - release time of job j, i.e. the earliest time at which job j can be scheduled,
- d_j - due date of job j, i.e. the time by which job j should be completed,
- w_j - weight (importance) of the job.

Upon constructing a schedule, based on the previous system properties, for each job the completion time C_j can be calculated, which in turns serves to define different scheduling objectives. The objective considered in this study is the total weighted tardiness (Twt), which can be defined as $TWT = \sum_j w_j \max(C_j - d_j, 0)$, and represents a weighted linear combination of the amount of time that each job spent executing after its respective due date. The problem under consideration can be classified as $R|r_j|TWT$ using the standard scheduling problem notation (Pinedo, 2012).

The above-described problem is considered under dynamic conditions. This means that no information about the jobs is known by the scheduler until they arrive. Naturally, the arrival times of jobs are also not known beforehand.

Only when the job is released, do all its properties (weight, processing times, due date) become known and can be used for scheduling. As a consequence, no information about the future of the system can be used when constructing the schedule, only the information about jobs that have already been released. Therefore, the whole schedule cannot be constructed in advance, but rather must be constructed in parallel with the execution of the system. This makes DRs the ideal choice for solving such dynamic problems, as they only make the next scheduling decision and do not schedule jobs in advance.

3.2. Designing DRs with GP

In general, DRs consist of two parts, the schedule generation scheme (SGS) and the priority function (PF). The SGS specifies how the entire schedule is constructed, meaning when the scheduling decisions should be performed and which job should be allocated to which machine. The outline of the SGS used by the automatically designed DRs is shown in Algorithm 1 (Đurasević & Jakobović, 2020). This SGS is invoked each time there is at least one unscheduled job in the system and at least one machine is free. At that point the SGS needs to determine which of the available jobs should be selected and scheduled on a machine. This is done by assigning a priority to each job-machine pair and then selecting the pair with the best priority value. The PF can be as simple as simply selecting the job-machine pair with the lowest processing time, or can be more complex as it is the case with the ATC rule (Đurasević & Jakobović, 2018). It should be outlined that the SGS calculates the priority for each machine, regardless of whether they are available or not. The reason for this is to introduce idle times in the schedule, i.e. to allow the SGS to determine that maybe another machine would be more suitable for a job than the one that is currently free. For example, another machine could soon become available and could process the considered job much faster, therefore it would be a waste to schedule the considered job on on a machine on which it would execute much longer.

From the previous description it is evident that the PF plays an important

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
        Get 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
```

role in the construction of schedules, as it is responsible for ranking the jobs based on which the SGS will select the next job to be scheduled. Therefore, it is important to use a good PF function which can provide a meaningful ranking of jobs, i.e. a ranking that will prioritise jobs that should be scheduled sooner in order to minimise the optimised criterion. Since designing new PFs manually is difficult, GP has often been applied to generate new and appropriate PFs for different problems. To apply GP for the generation of DRs it is required to specify the primitive set, i.e. the building blocks that will be used to construct the solution. The set of terminal nodes that are used is denoted in Table 1. This set consists both of simple system properties (processing time pt, weight w) as well as more complicated terminals that are calculated based on several system properties (slack of a job SL, average of processing times $pavg$). As for the function set, only simple operators are used because such a configuration lead to better results (Đurasević et al., 2016). The function nodes that are used are the summation, subtraction, multiplication, protected division (returns 1 if

Table 1: Terminal set

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

division by 0 is detected), and the unary positive operator ($pos(x) = \max(x, 0)$).

4. Ensembles of DRs

When considering ensembles of DRs, three design choices need to be considered. The fist choice is how the DRs in the ensemble should collaborate to construct the schedule. The second is which method will be used to construct the ensemble, i.e. select the individual DRs that will be contained in the ensemble. Finally, the size of the ensemble also needs to be specified.

4.1. Ensemble size

The size of the ensemble is an important parameter that directly influences its performance and execution time. Using a larger ensemble size usually leads to better solutions until a certain point at which the performance of the ensemble starts to deteriorate again. Larger ensembles are more costly to evaluate, but also much harder to interpret. As such, the goal is to keep the ensembles as small as possible.

4.2. Ensemble collaboration

In order to be able to apply the ensemble to a problem, it is required to define how the rules contained in the ensemble should collaborate to construct the solution. Until now, this has mostly been done using different ensemble combination methods, like summation and voting (Park et al., 2017). The idea behind these methods is that for each decision point in the system all rules in the ensemble are evaluated and the individual decisions are combined into a single decision which is then executed in the system. For example, in the case of the voting method, the job which was selected by most DRs will be scheduled next. The sum method, on the other hand, sums the priority values obtained by each DR for each decision, and the decision with the overall best priority value is executed. The benefits of these methods are that they are not expensive to calculate, but can easily outperform the results of individual DRs. An alternative to this is the ensemble collaboration method used for static scheduling, in which each rule in the ensemble constructs the entire solution and the best solution is selected and returned as the final solution. The main benefit of such ensembles is that they perform much better than individual rules, but are still very efficient to calculate. However, such a strategy is not applicable for dynamic scheduling environments, as not all information is available at the start of the system.

The ensemble collaboration method investigated in this paper, which was proposed in (Đurasević et al., 2022) and is denoted as parallel rule collaboration (PRC), is an attempt to adapt the latter ensemble collaboration method to dynamic scheduling problems. This collaboration method is based on the assumption that over time jobs get released into the system, and at one point several jobs will be waiting in the queue to be scheduled. The idea is that each rule in the ensemble independently simulates the scheduling of all currently released jobs and the value of the optimised criterion is calculated for each simulation. These simulations give a notion on how good each rule would perform if it would be used to schedule jobs that are currently waiting, with the assumption that no new jobs would arrive in the meantime. Based on these

simulations, the rule that obtained the best value of the optimised criterion for the simulation can be selected and used to perform the next scheduling decision for the real schedule. Algorithm 2 outlines the SGS used with the PRC collaboration method. It shows how in each scheduling decision all rules contained in the ensemble are applied to schedule all available jobs in a simulation. The rule which obtained the best criterion value for the simulated schedule is used to perform the next scheduling decision. This method slightly resembles the rollout heuristic(Đurasević & Jakobović, 2020), which uses a combination between exhaustive and heuristics search. The rollout procedure performs an exhaustive enumeration of all possibilities for the next immediate decision, i.e. which released job should be scheduled next. However, to determine which of these possibilities is the best one, it uses a simple heuristic rule to construct the rest of the schedule and obtain the quality of each of them. This is then used to determine which of the decisions in the exhaustive enumeration was the best, and execute in on the real schedule.

Algorithm 2: SGS used by PRC

```
1: while unscheduled jobs are available do
2:     Wait until at least one job and machine are available
3:     for each rule k in the ensemble E do
4:         Build the schedule only with released jobs at the current moment
5:         Store the quality of the simulated schedule in F[k]
6:     end for
7:     Determine the rule k_best which has the lowest value objective value of
       the simulated schedule based on their values in F[k]
8:     Apply rule k_best to determine the next scheduling decision
9: end while
```

The number of released jobs used in the simulation can be varied, which represents a configurable parameter of PRC and we denote as *look-ahead*. This parameter can take values between 1 and the current number of unscheduled

jobs released in the system. Using a smaller look-ahead value will result in a smaller computation time, whereas a larger parameter value will increase the computation time but will also improve its performance since the simulation will be performed by considering a larger number of jobs.

4.3. Ensemble construction

The ensemble construction methods are tasked with finding the best combination of DRs that should form the ensemble. All the algorithms tested in this study represent ensembles as integer arrays consisting of unique numbers, where each number represents the index of a DR from the set of available rules. Figure 1 outlines an example of such an encoding, in which five DRs are available for constructing ensembles, and an ensemble of size three needs to be constructed. In this case, the solution is represented as an array consisting of three numbers, which denote that the ensemble consists out of DRs with indices 1, 2, and 5. This ensemble is then interpreted using one of the previously described ensemble collaboration methods, concretely with the PRC method in this example. If at the current decision point ($t == 0$) six jobs are released in the schedule, each DR performs a simulation by scheduling all released jobs and calculating the criterion for such a solution. The DR that constructed the schedule with the lowest criterion value in its simulation is proclaimed the winner, and it is used to schedule the next job. In this case DR 5 achieved the lowest Twt value, meaning that it is used to determine which job should be scheduled at the current moment, in this case job 3. When job 3 completes with its execution, the ensemble is again used to determine the job that should be scheduled next.

Although various ensemble construction methods have been used in the literature, in this study we focus on using methods that construct ensembles from already generated DRs, as this methodology has shown a superior performance for the considered problem (Đurasević & Jakobović, 2017a). The motivation of this method is to divide the construction of ensembles from the generation of DRs, so that each method can solely focus on one part of the problem. Therefore, one method, like GP, is used to generate a set of DRs, after which another

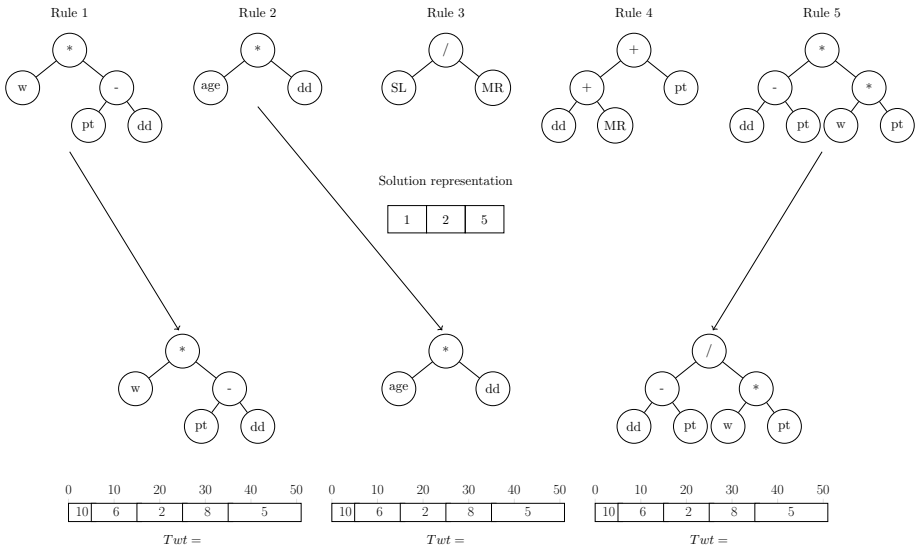


Figure 1: Ensemble representation and decoding

method is used to combine these rules into ensembles. One such method that uses existing DRs to construct ensembles is the simple ensemble construction (SEC), which is selected due to its good performance in previous studies Đurasević & Jakobović (2019). SEC uses a set of existing DRs and applies a strategy to determine which DRs from this set should be selected to form the ensemble. The main decision that needs to be performed when defining SEC is which strategy to use when selecting the rules for the ensemble. Therefore, various strategies were proposed, which will be used for constructing ensembles:

- Random - randomly select which DRs should form the ensemble
- Probabilistic - each rule has a probability proportional to its fitness of being selected
- Grow - the ensemble is built incrementally by selecting the rule that leads to the largest improvement in the performance of the ensemble when added to it
- Grow destroy - the ensemble is built in the same way as in the grow

method, but with twice the size. After it is constructed, rules are iteratively removed from it by removing those which lead to the largest improvement of the results until the desired ensemble size is reached

- Instance based - selects DRs which perform the best on the instances on which the currently constructed ensemble is achieving the worst results.

From the previous description it is evident that all strategies used by SEC are either based on random or greedy selection. However, the selection of the right combination of DRs that should form the ensemble is in itself an optimisation problem, and as such could also be tackled using evolutionary algorithms. This provides motivation to investigate whether better ensembles could be obtained by using a more sophisticated search procedure. For that purpose we propose a simple GA using the previously described integer representation. A simple crossover operator is used, which randomly selects indices (rules) from the parents and adds them to the child individual until the desired size. Therefore, an index that appears in both parents will have a higher probability of being transferred to the child than those that appear only once. During its execution, the operator ensures that no duplicate indices are placed into the child. The mutation operator, on the other hand, selects a random index in the individual and replaces it by a random index which is not yet contained in the individual. For parent selection and replacement the 3-tournament selection is used.

In addition to the aforementioned GA, we also adapt the memetic GA (MGA) from (Gil-Gala et al., 2020a). This algorithm uses a completely different intuition than all of the aforementioned methods. The previously described methods all rely on the calculation of the quality of the ensemble during their execution. However, the MGA does not evaluate the ensemble. It constructs the ensemble solely based on the fitness of individual DRs across all the problem instances. The MGA constructs the ensemble by selecting those rules that would achieve the best result across the entire problem set when each of them would be executed individually on each instance and the best result obtained by any of the rules would be selected for each instance. This method was used for

the offline scheduling problem, thus each rule in the ensemble would construct the solution and the best one would be selected. Regardless of this, we also apply MGA to construct ensembles that will be used with the ensemble collaboration methods applicable for dynamic problems. The MGA is implemented as a generational algorithm where parents are randomly selected for mating and a tournament between the parents and the obtained offspring is used to determine which individual advance to the next generation (thus ensuring elitism). The same encoding as in the aforedescribed GA is used to represent the individual. For the genetic operators the one point crossover is used, whereas the mutation is based on randomly changing between 1 and half of the elements in the ensemble. Finally, a local search algorithm is used as another genetic operator to improve some of the individuals of the population after they are evaluated. It uses a unique neighbourhood structure in which each neighbour is obtained by replacing the worst rule in the ensemble with another rule.

5. Experimental setup

In order to test the various ensemble collaboration and combination methods, a problem set of 180 instances was used. This set was divided into three disjoint sets, namely the training, validation, and test sets. Each of these sets contains 60 problem instances with different characteristics. The number of jobs ranges from 12 to 100, whereas the number of machines ranges between 3 and 10. Additionally, the instances were generated with different due date characteristics, meaning that certain instances will be easier whereas other will be harder to solve. More details about the problem instance generation procedure can be found in (Đurasević et al., 2016). The objective in these problem instances is to minimise the Total weighted tardiness (Twt) value. Since individuals are evaluated on sets consisting out of several instances, their fitness value is calculated as the sum of the Twt values obtained from the schedules constructed for each instance. In addition, since the sets contain instances of different sizes, the Twt values are additionally normalised to ensure that all the instances have a similar

influence on the total fitness value.

The training set is used by GP to evolve individual DRs. This is done in a way that GP was executed independently 50 times, and from each execution the best individual on the training set was stored. This resulted in a set of 50 DRs that are used to construct the ensembles. The GP used a population of 1 000 individuals, a mutation probability of 0.3, and a termination criterion of 80 000 function evaluations. In addition, the subtree, uniform, context-preserving, and size-fair crossover operators and subtree, hoist, node complement, node replacement, permutation, and shrink mutation operators were used (Poli et al., 2008). It should be outlined that all the rules used for this part were obtained in a previous study (Đurasević et al., 2016) and that no new rule generation was needed.

The validation set is used by the ensemble construction methods to select which rules should form the ensemble. The random and probabilistic strategies used by SEC will construct 1 000 random ensembles and select the best one. The GA uses a population of 100 individuals for PRC and 50 individuals for the sum and vote combination method, a mutation probability of 0.7, and 1 000 function evaluations. The MGA uses a population of 1 000 individuals, the crossover probability of 0.8, mutation probability of 0.2, and the local search is executed for 5 iterations with a ratio of 0.2 (it is applied on every fifth individual). It executes for 1 000 generations, which is more than the other methods, but again it needs to be stressed out that the fitness function of this method is different and not comparable to the others. All the aforementioned parameters were obtained after a preliminary parameter tuning phase.

Finally, the test set is used to evaluate the evolved ensembles on unseen problem instances. To validate the performance of the proposed PRC collaboration method, the summation and voting collaboration methods, which are most commonly applied in the literature, are used as a baseline for comparison Park et al. (2017); Đurasević & Jakobović (2019). All the results denoted in the tables and figures are calculated on this set. Each experiment is repeated 30 times to ensure statistical significance. In the results section several descriptive

statistics like the minimum, median, and maximum values will be denoted for the 30 obtained ensembles in each experiment. In addition, to check whether the results are statically significant, the Kruskal Wallis test is used with the post-hoc Dunn test. The results are considered significant if a p-value below 0.05 is obtained.

6. Results

In this section we denote the collaboration methods as sum (summation), vote (voting), and PRC-n, where n denotes the number of available jobs used in the simulation. PRC-∞ will be used to denote the case when all available jobs are used for the simulation. Regarding the ensemble construction methods, they will be denoted as rand (random), prob (probabilistic), grow (grow), grdt (grow destroy), and inst (instance based).

Table 2 shows the results obtained by each of the considered ensemble construction and collaboration methods. The best results for each ensemble collaboration method and each ensemble size are denoted with bold. As expected, it can be seen that no single ensemble construction method is dominant over all the case, which is verified with statistical tests. For PRC-1, no significant difference exists between the ensemble construction methods for size 3 (p-value=0.10). On the other hand for size 5 (p-value=0.0014) the rand method is significantly better than the grow, grdt and inst methods, and for size 7 (p-value$\approx$0) the inst method is significantly worse than any other ensemble construction method. In all other cases, the ensemble construction methods perform equally well, which can be seen from the table since the difference between their median values is small. For the PRC-∞ collaboration method and the ensemble size 3 (p-value$\approx$0) the MGA achieves statistically the best results, whereas for sizes 5 (p-value=0.153) and 7 (p-value=0.1) there are no significant differences between the construction methods. For the sum combination method, no significant difference exists for sizes 3 (p-value=0.12) and 5 (p-value=0.585) between the tested ensemble construction methods used. For size 7 (p-value$\approx$0) MGA performs significantly

worse than rand, grow, grdt, and inst, while GA performs significantly worse than inst. For the vote collaboration method, in case of size 3 (p-value=0.008) MGA performs better than grow and grdt, in case of size 5 (p-value≈0) MGA is significantly better than all methods except inst, and in case of size 7 (p-value≈0) MGA performs better than all other methods except GA and rand. In all cases we see that when compared to the individual rules, the constructed ensembles achieve better minimum, median and maximum values, which shows that they can easily outperform the individual rules.

Based on the previous analysis, it is difficult to outline which of the ensemble construction method would be the best. Therefore, for each parameter combination we rank the median values of each construction method (1 being the best), and then average the ranks across all parameter values to obtain the average rank that each construction method obtained. The average ranks, as well as the final rank of each method are denoted in Table 3. Additionally, the table also outlines the number of times the best median value was obtained for each construction method and ensemble size (denoted as NB in the table). These results show that certain methods tend to perform better than others when considering all the results. Most notably, MGA achieved the lowest rank, followed closely by rand and inst. On the other hand, GA, and the grow and grdt methods obtained the worst rank. Although these results are slightly surprising, they just show that simple methods already perform well enough, and that it is not required to use complicated optimisation algorithms to search for the best ensemble. Rather, it makes more sense to search for a better strategy by which the rules forming the ensemble should be selected, since based on the good results obtained by MGA and inst, this has a greater effect on the performance of the construction methods.

To better outline how the different ensemble construction methods perform for the novel PRC ensemble combination methods, Figure 2 shows the box plots for all three ensemble sizes. The box plots outline what was already demonstrated by statistical tests, which is that most methods achieve similar results. The one results that stands out most is the one obtained by MGA

Table 2: Results obtained by different ensemble construction and collaboration methods

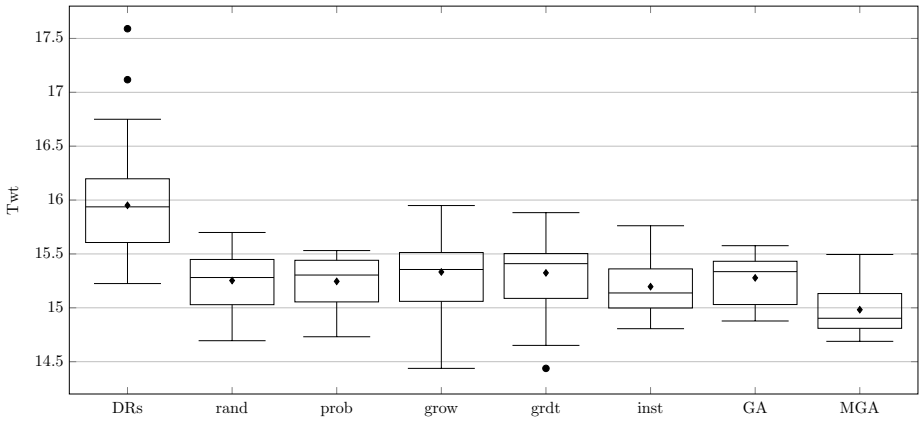
Collaboration	Construction	Size								
		3			5			7		
		min	med	max	min	med	max	min	med	max
	rand	14.86	**15.34**	**15.88**	15.01	**15.29**	**15.63**	**14.79**	15.44	15.88
	prob	15.03	15.42	15.93	14.91	15.42	16.08	15.08	15.43	16.01
	grow	**14.72**	15.47	16.18	15.06	15.44	16.11	14.89	15.45	15.98
PRC-1	grdt	14.95	15.48	15.99	14.96	15.42	16.19	15.01	15.49	**15.80**
	inst	14.95	**15.34**	15.92	15.09	15.50	15.95	15.27	15.63	16.11
	GA	14.90	15.43	16.02	14.82	15.41	16.12	15.03	15.49	16.00
	MGA	14.82	15.40	15.89	**14.79**	15.37	15.69	14.96	**15.35**	15.83
	rand	14.69	15.32	15.70	14.80	15.23	15.85	14.85	15.10	15.64
	prob	14.73	15.32	15.53	14.89	15.20	15.58	14.78	15.15	15.95
	grow	**14.44**	15.36	15.95	14.68	15.17	15.72	14.58	**15.12**	15.80
PRC-∞	grdt	**14.44**	15.41	15.88	**14.65**	15.17	15.82	14.78	15.14	15.73
	inst	14.81	15.15	15.76	14.69	**15.09**	**15.55**	**14.68**	15.15	15.66
	GA	14.87	15.35	15.58	14.73	15.24	15.88	14.76	15.25	16.16
	MGA	14.69	**14.92**	**15.49**	14.91	15.21	15.56	14.77	15.20	**15.52**
	rand	15.01	15.53	15.92	14.91	15.34	15.85	14.81	15.26	16.12
	prob	15.01	15.49	**15.80**	**14.73**	15.36	15.91	15.01	15.35	15.93
	grow	15.04	15.70	16.73	14.96	**15.27**	16.15	14.93	**15.20**	16.13
sum	grdt	15.03	**15.45**	16.63	14.95	15.32	16.15	14.93	15.24	16.05
	inst	**14.70**	15.51	16.08	14.87	15.33	15.88	14.93	15.22	15.88
	GA	14.86	**15.45**	16.10	15.02	15.38	15.97	**14.89**	15.36	16.03
	MGA	15.07	**15.45**	15.88	15.08	15.47	**15.79**	15.30	15.46	**15.87**
	rand	15.19	15.63	15.90	15.11	15.71	16.51	15.02	15.55	15.99
	prob	**14.98**	15.61	**15.87**	15.09	15.61	16.14	15.04	15.73	16.55
	grow	15.07	15.69	16.40	15.19	15.70	16.40	**14.82**	15.70	16.22
vote	grdt	15.07	15.73	16.40	15.12	15.69	16.40	15.08	15.65	16.22
	inst	15.14	15.73	16.29	**14.99**	15.53	16.46	15.07	15.66	16.02
	GA	15.08	15.57	16.27	15.13	15.61	**16.09**	15.11	15.58	16.22
	MGA	15.10	**15.49**	16.03	15.08	**15.41**	16.12	15.08	**15.41**	**15.78**
Individual rules		15.23	15.94	17.59	15.23	15.94	17.59	15.23	15.94	17.59

Table 3: Collective statistics for different ensemble construction methods

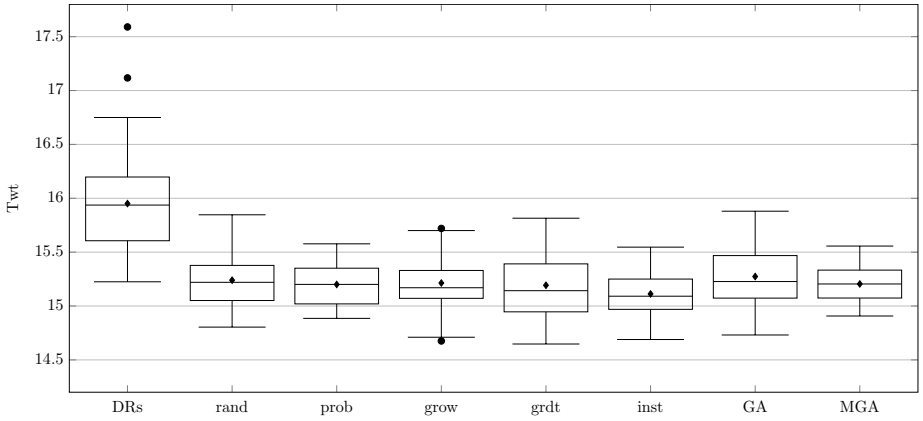
Construction method	NB	Avg. rank	Final rank
MGA	6	3	1
rand	3	3.50	2
inst	2	3.75	3
prob	0	4.08	4
grdt	1	4.17	5
grow	2	4.33	6
GA	0	4.58	7

for ensemble size 3, which represents the best result from all the experiments and is significantly better than all others. Furthermore, it can be seen that the performance for most methods increases with the increase of the ensemble size. In all considered cases the constructed ensembles perform significantly better than the individual DRs. Even more, for the larger ensemble sizes it can be seen that even up to 50% of the obtained ensembles achieve a better performance than the best individual DRs, which additionally speaks in favour of their performance.

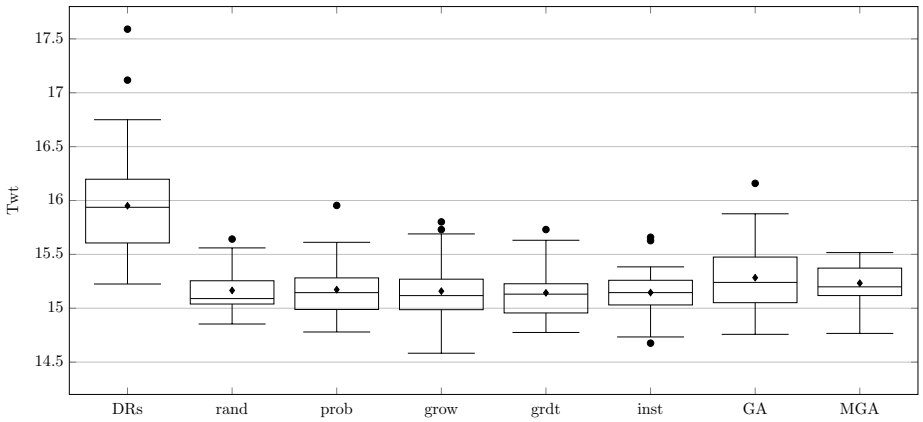
When comparing the collaboration methods (using the construction method which achieved the best median value) for the considered ensemble sizes, the differences are much more evident. For ensembles of size 3 the collaboration methods achieve significantly different results (p-value$\approx$0), with PRC-∞ performing significantly better than all the other methods, and PRC-1 performing significantly better than vote. For size 5 the results are again significantly different (p-value$\approx$0), with PRC-∞ achieving significantly better results than other collaboration methods. Finally for size 7 a significant difference can again be observed (p-value$\approx$0) with PRC-∞ performing better than any other collaboration method, and sum performing better than vote. These results demonstrate that the proposed PRC-∞ always significantly outperforms the sum and vote



(a) Ensemble size 3



(b) Ensemble size 5



(c) Ensemble size 7

Figure 2: Performance of the PRC-∞ collaboration method with ensembles constructed by various construction methods

Table 4: Collective statistics for different ensemble collaboration methods

	NB	Avg. rank	Final rank
PRC-1	1	2.52	2
PRC-∞	20	1.05	1
sum	0	2.57	3
vote	0	3.86	4

collaboration methods that were mostly considered in the literature. Similarly as for the ensemble construction methods, we denote the average and final rank, as well as the number of best achieved median values for each ensemble collaboration method in Table 4. It can clearly be seen that the PRC-∞ method achieved the best median value in all experiments except one. On the other hand, the sum and PRC-1 methods have a similar average rank, and as such can be considered equally efficient. This suggests that using only a single job for the simulation might is enough to outperform the standard ensemble combination methods, but rather more jobs need to be considered to get a significant improvement.

To better outline the differences between the various ensemble collaboration methods, the best result for each of the collaboration methods is shown in Figure 3 for the three considered ensemble sizes. The figure clearly shows that PRC-∞ constantly obtained much better results than the other methods, whereas the performance of the other methods depends highly on the size of the ensemble that was used. However, all methods clearly outperformed individual DRs by a large margin.

7. Further analysis

In this section, we further analyse different elements of the proposed PRC method and make more detailed comparisons between the different ensemble construction and collaboration methods.

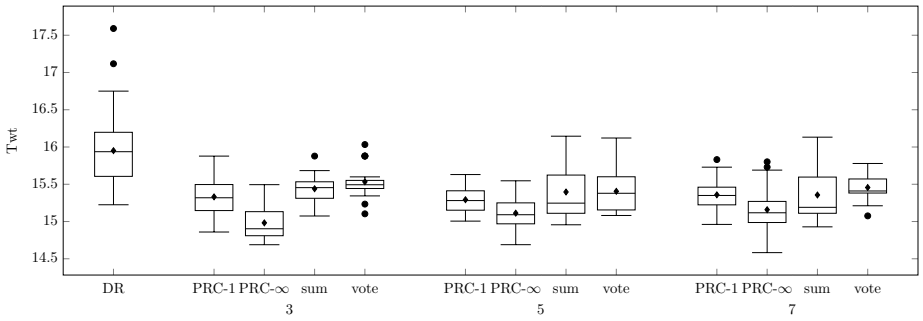
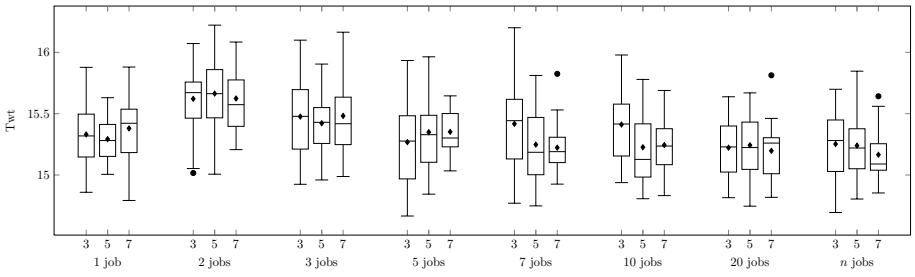


Figure 3: Comparison of the best results obtained by each collaboration method

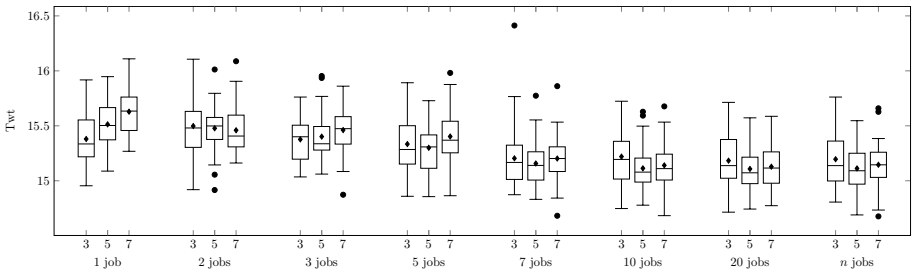
7.1. Influence of the look-ahead parameter

In this section we investigate the influence the look-ahead parameter of the PRC method. This parameter determines how many unscheduled jobs that are released into the system will be used in the simulations created by the DRs of the ensemble to determine which rule performs the best decision. Naturally, it is expected that as more jobs are considered in the look-ahead that the simulations will be more precise. This will allow the ensemble to select the rules that performs a better decision at the current point, which should result in an overall better performance of the ensemble. The method was tested with several values for this parameter and the results are summarised in Figure 4. The tests were performed using the rand, inst and MGA methods as they achieved the best results.

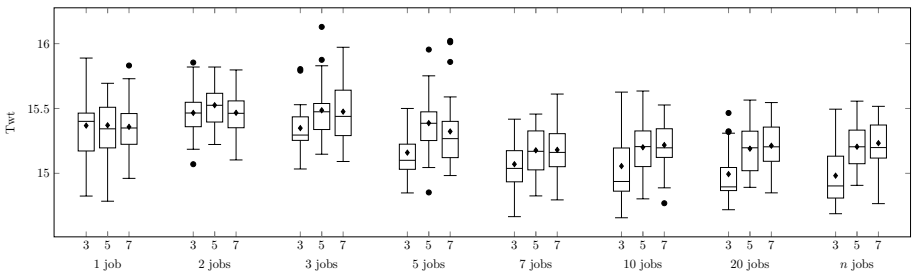
For all three methods a similar behaviour can be observed for the tested look-ahead parameter values. As for the look-ahead parameter, it can be seen that for smaller values (1-3) the method performs worse and is slightly unstable. For example, for some cases (especially for the rand method) it can happen that the increase in the look-ahead parameter does not lead to a better performance. The reason for this is that it is difficult to perform good approximations based only on a few jobs in the simulation. Thus a small variation in the look-ahead parameter value can largely affect the behaviour of the ensemble. However, as more jobs are used in the simulation, the results become more consistent,



(a) Results obtained by the rand method



(b) Results obtained by the inst method



(c) Results obtained by the MGA

Figure 4: Results for different values for the look-ahead parameter of PRC

especially when 7 or more jobs are used. It can even be seen that with smaller look-ahead parameter values in some cases it is possible to achieve similar results to those when all released jobs are considered. Thus the execution time of the method can be improved with no or a small decrease in the performance of ensembles.

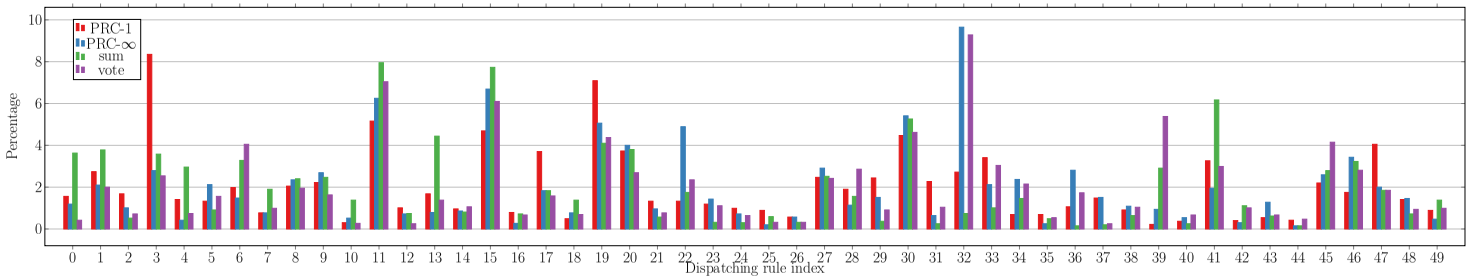
7.2. Composition of ensembles

An interesting thing which we want to investigate is the composition of ensembles. We want to analyse whether any particular ensemble combination or

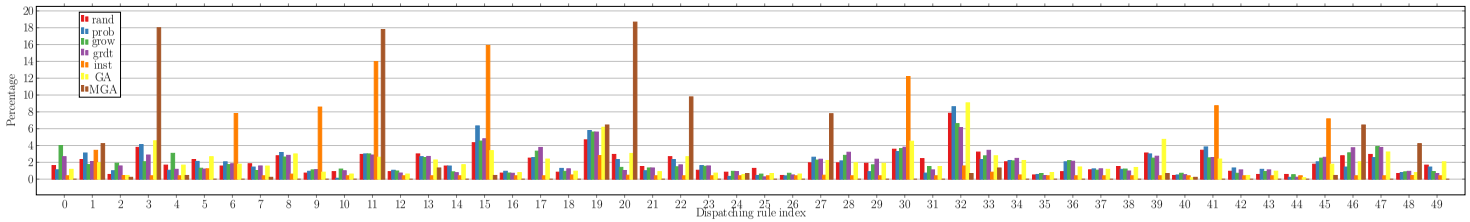
construction method has a preference to select certain rules into the ensemble. Figure 5 shows a histogram of the percentages that each of the 50 considered DRs appeared in the best ensembles constructed by each of the methods. First, subfigure 5a shows the histogram for the considered ensemble collaboration methods. We observe that regardless of the collaboration method that is used, all of them have similar preferences. For example, all collaboration methods quite often use rules with indices 11, 15, 19, and 30 for constructing the ensembles. On the other hand, there is also a group of rules that are rarely used by any of the collaboration methods, such as rules with indices 2, 10, 12,14, 16, 18, 23-26, 35, 40, 42-44. This shows that a lot or rules simply do not seem to be suitable for being used in the ensembles. Additionally, in some cases certain rules are used more often by specific collaboration methods, like in the case of rule with index 3 that is mostly used by PRC-1, or rule 32 which is usually used by PRC-∞ and vote. However, it is very interesting that a lot of rules are often selected for all collaboration methods, which would suggest that the selection of the rules for the ensemble is not heavily dependent on the collaboration method which.

Subfigure 5b shows the distribution of rules used by different ensemble construction methods. In this case, more evident differences can be observed. Again, there are several DRs that are not selected by any ensemble construction method, which are the same as those outlined previously. However, for those ensembles which are selected most of the times it can be noticed that there is discrepancy between the different methods. Most notably, the inst and MGA methods usually select certain rules more often than the other methods, whereas the other five methods behave more consistently with each other. However, this is expected, as both MGA and inst use a completely different strategy to select the rules that form the ensemble in comparison to the other methods.

Based on the previous analysis, we can conclude that some rules are inherently more favoured to be selected to form ensembles, regardless of which ensemble combination or construction method is used. This suggests that some DRs are a priori more suitable to be used for ensembles. However, the question



(a) Distribution of selecting rules by different ensemble construction method



(b) Distribution of selecting rules by different ensemble construction methods

Figure 5: The distributions of DRs contained in the best obtained ensembles

is whether such a thing could be detected from those rules individually, without constructing and evaluating ensembles. If yes, this could lead to the design of methods that could construct general ensembles which could be used by any collaboration method in a much shorter time. One potential feature that could be used for that is the individual performance of DRs. Figure 6 shows the dependency between the individual fitness of DRs and the percentage by which they appear in the best ensembles obtained by all collaboration and construction methods. This figure suggests that the most fit rules appear more often in ensembles. However, it can also be seen that some quite poor rules appear in the same percentage as some more fit rules. Thus, it seems that it is not possible to solely use fitness for this decision. This is also backed up by the performance of the probabilistic construction method which favoured the selection of more fit individuals, but that did not lead to any improvement in the results. To gain a better notion on whether any correlation between the individual fitness and percentage of it appearing the the best ensembles exists, the Spearman's Rho correlation test was used. The correlation value of -0.414 was obtained, which suggest a slight negative correlation. This partially backs up the previous observations, but still shows that because of the lack of a stronger correlation between these two measures we would require additional information by which to determine the suitability of DRs for ensembles.

7.3. Generality of rules

In this section we further investigate whether ensembles constructed for one collaboration method are reusable by different collaboration methods. This analysis is performed by interpreting the ensembles constructed with one collaboration methods with the remaining three collaboration methods. Table 5 shows the summary of median values obtained for the ensembles of size 5, since for this size most of the collaboration methods achieved the best overall results. The methods denoted in the rows represent the collaboration method used during learning, whereas the method denoted in the column is the one used to interpret the constructed ensemble. The difference between the obtained result

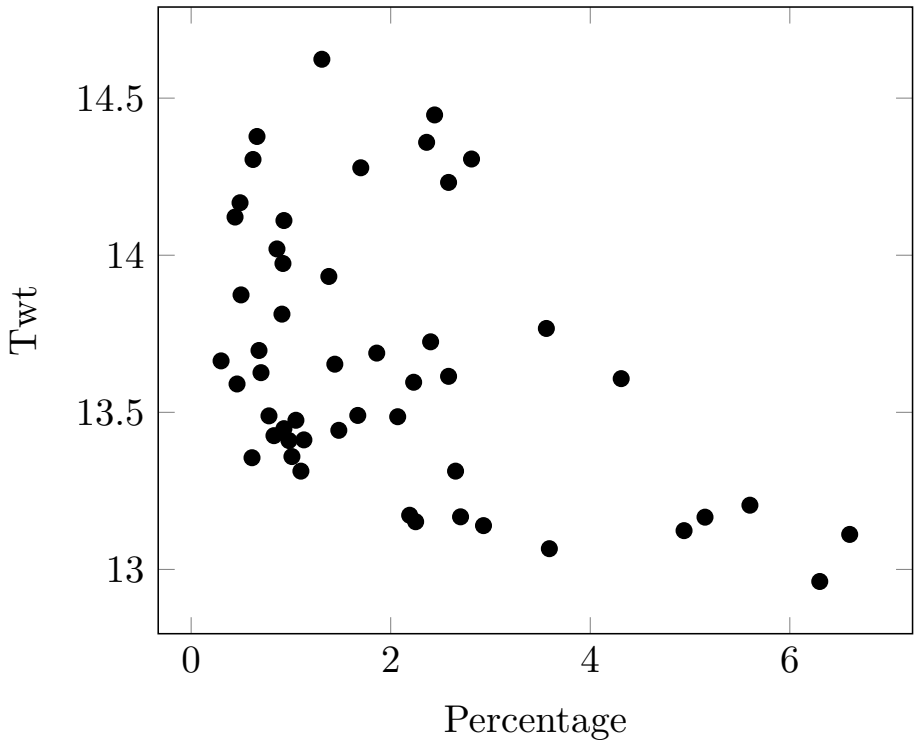


Figure 6: Scatter plot denoting the dependency between the fitness of rules and the percentage by which they appear in ensembles

and the reference median value obtained when the same collaboration method was used both during ensemble construction and evaluation is denoted in brackets. A negative value denotes that the obtained value is lower than the reference value (meaning the results improved), whereas a positive value denotes it is larger (meaning the results deteriorated). The table shows that certain collaboration methods are more resilient. For example, ensembles constructed with any collaboration methods can be interpreted with PRC-∞ or vote and the results will be similar to those that were obtained if those two methods would have been used during the ensemble construction process. This can best be seen from the fact that in both cases the results do not deteriorate more than by 1% for all cases. The situation with PRC-1 is slightly different, as the results are somewhat worse when ensembles constructed for other collaboration methods are interpreted as PRC-1 ensembles. However, the sum collaboration method is the least general, meaning that when using it on ensembles constructed by other collaboration methods, the results are in many cases significantly worse. The deterioration of the results in comparison to the reference median values can even reach up to 3%. Looking from the side of the methods, the inst and MGA methods seem to obtain ensemble sets which work generally the best for all collaboration methods. Since MGA evolves the same ensemble regardless of the collaboration method, the results only depend on the collaboration method that is used to interpret them.

These observations are quite important as they show several things. First, they give additional proof that similar rules are used by different ensemble collaboration methods. Therefore, one ensemble can in most cases be efficiently interpreted with an alternative ensemble collaboration method. Second, the PRC-∞ method demonstrated to be the most resilient method to these changes, which means that the ensembles can be constructed using a more computationally efficient collaboration method, and then this ensemble can later be interpreted with CRP-∞.

To further investigate the generality of the PRC method, we investigate how the ensembles evolved for the PRC-1 and PRC-∞ perform when using different

Table 5: Results obtained when interpreting ensembles obtained for one collaboration method with other collaboration methods

		Interpretation			
		PRC-1	PRC-∞	sum	vote
PRC-1	rand	15.29	15.19 (-0.01)	15.47 (+0.13)	15.67 (-0.04)
	prob	15.42	15.33 (+0.16)	15.77 (+0.41)	15.57 (-0.04)
	grow	15.44	15.17 (0)	15.67 (+0.40)	15.68 (-0.02)
	grdt	15.42	15.16 (-0.01)	15.75 (+0.43)	15.68 (-0.01)
	inst	15.50	15.04 (-0.05)	15.30 (-0.03)	15.57 (+0.04)
	GA	15.41	15.39 (+0.15)	15.67 (+0.29)	15.73 (+0.12)
	MGA	15.37	15.21 (0)	15.47 (0)	15.41 (0)
PRC-∞	rand	15.64 (+0.35)	15.20	15.74 (+0.40)	15.69 (-0.02)
	prob	15.50 (+0.08)	15.17	15.70 (+0.34)	15.77 (+0.16)
	grow	15.50 (+0.06)	15.17	15.70 (+0.43)	15.77 (+0.07)
	grdt	15.50 (+0.08)	15.17	15.84 (+0.52)	15.80 (+0.11)
	inst	15.42 (-0.08)	15.09	15.39 (+0.06)	15.54 (+0.01)
	GA	15.50 (+0.09)	15.24	15.73 (+0.35)	15.69 (+0.08)
	MGA	15.37 (0)	15.21	15.47 (0)	15.41 (0)
sum	rand	15.49 (+0.20)	15.14 (-0.06)	15.34	15.74 (+0.03)
	prob	15.56 (+0.14)	15.19 (+0.02)	15.36	15.65 (+0.04)
	grow	15.62 (+0.18)	15.20 (+0.03)	15.27	15.59 (-0.11)
	grdt	15.56 (+0.14)	15.22 (+0.05)	15.32	15.65 (-0.04)
	inst	15.59 (+0.07)	15.17 (+0.08)	15.33	15.63 (+0.10)
	GA	15.46 (+0.05)	15.22 (-0.02)	15.38	15.71 (+0.10)
	MGA	15.37 (0)	15.21 (0)	15.47	15.41 (0)
vote	rand	15.66 (+0.37)	15.11 (-0.09)	15.67 (+0.33)	15.71
	prob	15.69 (+0.27)	15.13 (-0.04)	15.68 (+0.32)	15.61
	grow	15.51 (+0.07)	15.18 (+0.01)	15.62 (+0.35)	15.70
	grdt	15.50 (+0.08)	15.15 (-0.02)	15.60 (+0.28)	15.69
	inst	15.52 (+0.02)	15.14 (+0.05)	15.25 (-0.08)	15.53
	GA	15.51 (+0.10)	15.11 (-0.13)	15.73 (+0.35)	15.61
	MGA	15.37 (0)	15.21 (0)	15.47 (0)	15.41

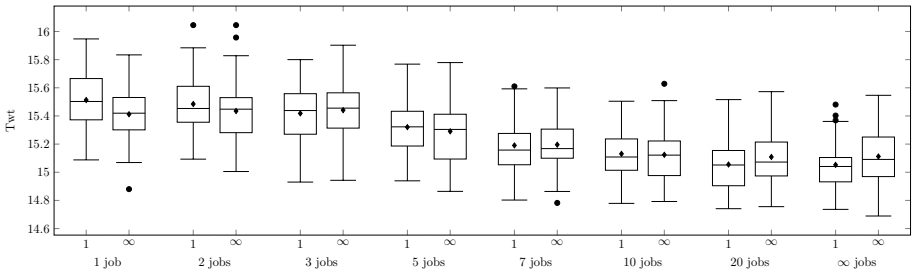


Figure 7: Results for applying the PRC-1 and PRC-∞ methods with different look-ahead parameters

number of jobs for the look-ahead parameter. Figure 7 shows the box plot of the results for PRC-1 (denoted as 1 in the image) and PRC-∞ (denoted as ∞ in the image) across different look-ahead values. The results demonstrate that regardless which collaboration method was used when constructing the ensembles, both work equally well when used with different look-ahead values. This observation is important, as it shows that the lowest look-ahead value can be used during construction of the ensembles to speed up the learning process. Then, when required, this parameter can freely be varied to larger values without any performance loss.

7.4. Runtime analysis

Table 6 outlines execution times in seconds obtained by all ensemble collaboration methods for the considered ensemble sizes. These execution times represent the average of executing 30 ensembles for each method when solving all instances in the test set. In addition, the execution time obtained for the individual DRs, which is equal to 0.08 seconds, is also outlined. As can be seen, the vote and sum method have a quite similar execution time, which is only slightly larger than the execution time of the individual DRs. On the other hand, the execution times of the PRC method are significantly higher. Even when only a single job is considered in the simulation, the method is slower by a factor similar to the number of rules in the ensemble. As the number of jobs in the simulation increases, the execution times increase, however, the increase

Table 6: Execution times in seconds of different ensemble collaboration methods

Size	Method										
	DR	sum	vote	PRC-1	PRC-2	PRC-3	PRC-5	PRC-7	PRC-10	PRC-20	PRC-∞
3	0.08	0.12	0.10	0.25	0.42	0.53	0.71	0.85	0.95	1.01	1.00
5	0.08	0.18	0.15	0.39	0.66	0.90	1.20	1.43	1.62	1.96	1.95
7	0.08	0.23	0.21	0.53	0.91	1.19	1.64	1.97	2.47	2.60	2.80

is smaller as more jobs are considered. The reason for this is that there are less situations in the system when such a large number of unscheduled jobs is available. In the end, for a reasonable number of jobs in the simulation (7 or higher) the method is about an order of magnitude slower than the individual DRs, equalling to a few seconds. Although even these values should be acceptable for dynamic systems, it should be noted that this is the execution time for the entire test set, therefore individual scheduling decisions are performed in a much smaller time.

Table 7 shows the execution times for different ensemble construction methods. The sum combination method was used during measurement, but based on the ratios between the execution times of collaboration methods it would be easy to approximate their execution times also for the other collaboration methods. As can be seen, the rand, prob and GA methods have a similar execution times for all ensemble sizes, which is expected as they were fixed to evolve 1000 ensembles. This just shows that using additional probability calculations in the prob method, or genetic operators in GA does not lead to any significant overhead. The execution times for the grow and grdt methods are denoted for constructing only a single ensemble. Therefore it can be seen that the execution times for constructing only a single ensemble are quite high. The reason for this is that they greedily construct the schedule and therefore perform many ensemble evaluations. However, as seen by the results, they did not perform better than the faster and more simple rand method. Finally, inst and MGA have the lowest execution times, although they achieve among the best results. The reason is that MGA does not use the evaluation of ensembles at all, while

Table 7: Execution times in seconds of different ensemble construction methods

Size	Method						
	rand	prob	grow	grdt	inst	GA	MGA
3	116.50	115.00	9.20	32.68	0.33	105.53	7.89
5	168.00	167.00	22.71	80.71	0.71	156.40	12.87
7	222.90	219.50	40.17	149.54	1.21	207.00	14.01

inst does only each time a new rule is added to the ensemble. This makes these two methods the most efficient.

7.5. Ensemble size analysis

A valid question that can be raised is whether the results could be further improved if larger ensemble sizes would be used aside from those that were used for the main experiments. Figure 8 shows the median values obtained by the collaboration methods for different ensemble sizes. As can be seen from the results, using larger ensemble sizes does not bring any improvement over smaller ensembles. On the contrary, the results of PRC-1 and sum deteriorate significantly with a too large increase of the ensemble size. PRC-∞ and vote seem to be more resilient to the number of rules in the ensemble and even obtain some good results for certain larger sizes. This finding is important as it motivates to keep ensembles smaller, which also has a positive effect on their runtime.

7.6. Analysis on the frequency of using rules in the ensemble

In this section we shortly analyse how many times each rule in the ensembles generated for the PRC-∞ collaboration method are actually selected to perform the decisions. The method used for the construction of the ensemble was the instance based method. The reason for selecting this method was due to its property that larger ensembles always contain the same DRs as the smaller ones, therefore it is easier to denote the change in the application percentage of

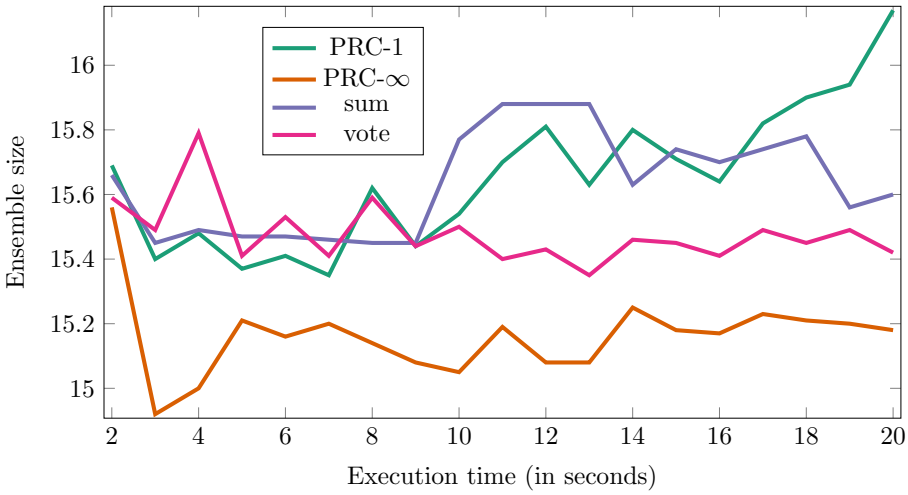


Figure 8: Median values obtained for different ensemble sizes when using the ensembles constructed by MGA

DRs as the ensemble size increases. However, similar distributions were observed also for other ensemble construction and collaboration methods.

Table 8 provides the summary for three selected rules, one for each of the considered ensemble sizes. The table denotes the individual fitness of each rule in the ensemble, and for each rule the percentage of decisions in which this rule was applied. As it can be seen, usually one rule is applied in most situations. For ensemble size 3 this rule is applied in around 80% of situations, whereas for the ensembles of sizes 5 and 7 it is applied in around 76% of cases. For the other rules it can again be seen that one or two rules are usually applied in more situations than others. Especially for size 7, some rules are applied only in around 2% of decisions. Looking at the fitness of the rules contained in the ensembles, one can notice that rules of different quality are selected to form the ensemble, from worse (rules 45 and 46), to better ones (rule 33). Even more, one of the worst rules in the ensemble (rule 46) is actually used to perform most of the decisions. Regardless of this, the ensembles in the end achieve a significant improvement in the performance in comparison to the individual rules. These results demonstrate how the rules can complement each other, since one rule

Table 8: Frequency of applying individual DRs in the ensemble used by PRC-∞

	Ensemble size	Rule 46	15	30	11	1	33	45	Ensemble fitness
Rule fitness		16.02	15.77	15.67	15.72	15.92	15.31	16.23	
Application percentage	3	80.28	14.20	5.52	-	-	-	-	14.80
	5	76.55	5.97	3.38	10.90	3.20	-	-	14.69
	7	76.43	5.57	2.96	8.73	2.20	2.42	1.70	14.68

that generally performs good decisions can be used in most of the cases, and with other rules being applied to more specific situations to improve the results. Furthermore, this analysis also shows that even poor rules can be improved by complementing them with additional rules that make better decision in certain situations.

7.7. Influence of the set used for ensemble construction

The final thing we analyse is whether there is really a need for using the validation set, or can the same training set that was used to generate DRs also be used to construct the ensembles. For that purpose, the main experiments were performed once again, however, this time by using the training set during the construction of the ensembles. Table 9 shows only a short summary of the results when comparing both sets. The table outlines the number of times better ensembles were obtained using the validation or training set, as well total difference between their median values (a negative value denotes that better median values were obtained using the validation set in addition). The results denote that it is not easy to say that using either set is better. Certain construction methods seem to work better with a certain set, although in more cases better median values were obtained when using the validation set to construct the ensembles. Thus it might seem to be slightly better to use the additional validation set, since it might reduce the chance of overfitting the ensembles to the problems that were already used for designing DRs. However,

Table 9: Summary of the results when using the validation or the training set for constructing ensembles

	Validation better	Training better	Median difference
rand	10	2	-0.83
prob	8	4	-0.80
grow	7	5	-0.15
grdt	3	9	0.04
inst	5	7	0.50
GA	7	5	-0.24
MGA	6	6	0.17

even if the training set is used to construct the ensembles, it will not lead to a large difference in the results.

8. Conclusion

This paper investigates the application of a novel ensemble collaboration method based on a different premise than standard methods that combine the decisions of individual rules into a single one. The paper also investigates two novel applications of evolutionary algorithms to construct ensembles. The results demonstrated that the proposed ensemble collaboration method achieves a significantly better performance than standard collaboration methods used in the literature. Furthermore, it was demonstrated that the performance of proposed collaboration method was not largely affected by the choice of rules in ensemble or the look-ahead parameter, which outlines its resilience to the choice of parameters. Naturally, the proposed collaboration method does have certain limitations that need to be taken into account. The execution time of this method is larger than those of the other standard collaboration methods like vote and sum, although still being small enough to be applicable in dynamic environments. Furthermore, this collaboration method is only applicable

in overloaded systems, since it is required that a certain number of jobs are waiting to be scheduled at each point in time so that the ensemble can construct a simulation of the schedule.

The additional analysis of the obtained results show several interesting findings. First, it is demonstrated that algorithms which rarely evaluate ensembles, but perform the selection of the rules only based on their individual performance, achieve equally good results as the more expensive ones, but in a smaller execution time. In addition, the analyses demonstrate that all ensemble collaboration methods in most cases use similar rules to construct ensembles. This shows that some rules are inherently better suited to form ensembles than others. As a result, an ensemble of rules could be constructed regardless of which collaboration method is used to interpret it, meaning that only a single ensemble needs to be constructed and can be interpreted using different collaboration methods. All these findings provide interesting insights in hyper-heuristic ensembles, which show that certain design choices in the construction of ensembles need to be reevaluated.

Based on the obtained results, we identify three main future research directions. First, we plan to research alternative ensemble combination methods to investigate whether it is possible to discover better ways in which rules in the ensemble could collaborate. Furthermore, it would be interesting to investigate the combination of both collaboration methods, i.e. creating ensembles of ensembles. This could potentially improve the results at the cost of a slower execution time. Finally, as the MGA and the instance based strategy for SEC achieved among the best results, this provides more motivation to research ensemble constructing methods which are not guided by the performance of the constructed ensemble, but rather by certain properties of the individual rules.

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.

References

Anagnostopoulos, G., & Rabadi, G. (2002). A simulated annealing algorithm for the unrelated parallel machine scheduling problem. In *Proceedings of the 5th Biannual World Automation Congress* (pp. 115–120). volume 14. doi:10.1109/WAC.2002.1049430.

Ardeh, M. A., Mei, Y., & Zhang, M. (2021). Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, (pp. 1–1). doi:10.1109/TEVC.2021.3129278.

Arnaout, J.-P., Musa, R., & Rabadi, G. (2012). 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*, 43–53. URL: https://doi.org/10.1007/s10845-012-0672-3. doi:10.1007/s10845-012-0672-3.

Bhowan, U., Johnston, M., Zhang, M., & Yao, X. (2013). Reusing Genetic Programming for Ensemble Selection in Classification of Unbalanced Data. *IEEE Transactions on Evolutionary Computation*, .

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

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

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, *64*, 1695–1724. URL: https://doi.org/10.1057/jors.2013.71. doi:10.1057/jors.2013.71.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2010). A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics* (pp. 449–468). Springer US. URL: https://doi.org/10.1007/978-1-4419-1665-5_15. doi:10.1007/978-1-4419-1665-5_15.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R. (2018). A classification of hyper-heuristic approaches: Revisited. In *International Series in Operations Research and Management Science* (pp. 453–477). Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-91086-4_14. doi:10.1007/978-3-319-91086-4_14.

Chand, S., Huynh, Q., Singh, H., Ray, T., & Wagner, M. (2018). On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems. *Information Sciences*, *432*, 146 – 163. URL: http://www.sciencedirect.com/science/article/pii/S0020025517311350. doi:https://doi.org/10.1016/j.ins.2017.12.013.

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

Fan, Q., Bi, Y., Xue, B., & Zhang, M. (2022). Evolving effective ensembles for image classification using multi-objective multi-tree genetic programming. In H. Aziz, D. Corrêa, & T. French (Eds.), *AI 2022: Advances in Artificial Intelligence* (pp. 294–307). Cham: Springer International Publishing.

Fanjul-Peyro, L., & Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational*

Research, *207*, 55–69. URL: https://doi.org/10.1016/j.ejor.2010.03.030. doi:10.1016/j.ejor.2010.03.030.

Folino, G., Pizzuti, C., & Spezzano, G. (2005). Gp ensemble for distributed intrusion detection systems. In S. Singh, M. Singh, C. Apte, & P. Perner (Eds.), *Pattern Recognition and Data Mining: Third International Conference on Advances in Pattern Recognition, ICAPR 2005, Bath, UK, August 22-25, 2005, Proceedings, Part I* (pp. 54–62). Berlin, Heidelberg: Springer Berlin Heidelberg.

Folino, G., Pizzuti, C., & Spezzano, G. (2008). Training distributed gp ensemble with a selective algorithm based on clustering and pruning for pattern classification. *Trans. Evol. Comp*, *12*, 458–468.

Ganaie, M., Hu, M., Malik, A., Tanveer, M., & Suganthan, P. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, *115*, 105151. URL: https://doi.org/10.1016%2Fj.engappai.2022.105151. doi:10.1016/j.engappai.2022.105151.

Gedik, R., Kalathia, D., Egilmez, G., & Kirac, E. (2018). A constraint programming approach for solving unrelated parallel machine scheduling problem. *Computers & Industrial Engineering*, *121*, 139–149. URL: https://www.sciencedirect.com/science/article/pii/S0360835218302158. doi:https://doi.org/10.1016/j.cie.2018.05.014.

Gil-Gala, F. J., Mencía, C., Sierra, M. R., & Varela, R. (2019). Evolving priority rules for on-line scheduling of jobs on a single machine with variable capacity over time. *Applied Soft Computing*, *85*, 105782. URL: https://www.sciencedirect.com/science/article/pii/S1568494619305630. doi:https://doi.org/10.1016/j.asoc.2019.105782.

Gil-Gala, F. J., Mencía, C., Sierra, M. R., & Varela, R. (2020a). Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms. *Integrated Computer-Aided Engineering*, *28*, 65–80. URL: https://doi.org/10.3233/ICA-200634. doi:10.3233/ICA-200634.

Gil-Gala, F. J., Sierra, M. R., Mencía, C., & Varela, R. (2020b). Combining hyper-heuristics to evolve ensembles of priority rules for on-line scheduling. *Natural Computing*, . URL: https://doi.org/10.1007/s11047-020-09793-4. doi:10.1007/s11047-020-09793-4.

Gil-Gala, F. J., Sierra, M. R., Mencía, C., & Varela, R. (2021). Genetic programming with local search to evolve priority rules for scheduling jobs on a machine with time-varying capacity. *Swarm and Evolutionary Computation*, *66*, 100944. URL: https://www.sciencedirect.com/science/article/pii/S2210650221001061. doi:https://doi.org/10.1016/j.swevo.2021.100944.

Gil-Gala, F. J., & Varela, R. (2019). 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* (pp. 223–233). Springer International Publishing. URL: https://doi.org/10.1007/978-3-030-19651-6_22. doi:10.1007/978-3-030-19651-6_22.

Hart, E., Ross, P., & Corne, D. (2005). Evolutionary scheduling: A review. *Genetic Programming and Evolvable Machines*, *6*, 191–220. URL: https://doi.org/10.1007/s10710-005-7580-7. doi:10.1007/s10710-005-7580-7.

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

Iba, H. (1999). Bagging, boosting, and bloating in genetic programming. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2* GECCO'99 (pp. 1053–1060). Morgan Kaufmann Publishers Inc.

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

Jaklinović, K., Đurasević, M., & Jakobović, D. (2021). Designing dispatching rules with genetic programming for the unrelated machines environment with constraints. *Expert Systems with Applications*, *172*, 114548. URL: https://www.sciencedirect.com/science/article/pii/S0957417420311921. doi:https://doi.org/10.1016/j.eswa.2020.114548.

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

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

Muñoz, L., Silva, S., & Trujillo, L. (2015). M3gp – multiclass classification with gp. In P. Machado, M. I. Heywood, J. McDermott, M. Castelli, P. García-Sánchez, P. Burelli, S. Risi, & K. Sim (Eds.), *Genetic Programming* (pp. 78–91). Cham: Springer International Publishing.

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

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2013). A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation*, *17*, 621–639.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2013). Dynamic multi-objective job shop scheduling: A genetic programming approach. In *Studies in Computational Intelligence* (pp. 251–282). Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-39304-4_10. doi:10.1007/978-3-642-39304-4_10.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2018). Genetic programming for job shop scheduling. In *Studies in Computational Intelligence* (pp. 143–167). Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-91341-4_8. doi:10.1007/978-3-319-91341-4_8.

Ochoa, G., Vazquez-Rodriguez, J. A., Petrovic, S., & Burke, E. (2009). Dispatching rules for production scheduling: A hyper-heuristic landscape analysis. In *2009 IEEE Congress on Evolutionary Computation*. IEEE. URL: https://doi.org/10.1109/cec.2009.4983169. doi:10.1109/cec.2009.4983169.

Paris, G., Robilliard, D., & Fonlupt, C. (2001). Applying Boosting Techniques to Genetic Programming. *Artificial Evolution 5th International Conference, Evolution Artificielle, EA 2001*, *2310*, 267–278. doi:doi:10.1007/3-540-46033-0_22.

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

Park, J., Mei, Y., Nguyen, S., Chen, G., & Zhang, M. (2017). 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.

Park, J., Nguyen, S., Zhang, M., & Johnston, M. (2015). Evolving ensembles

of dispatching rules using genetic programming for job shop scheduling. (pp. 92–104). doi:10.1007/978-3-319-16501-1_8.

Pinedo, M. L. (2012). *Scheduling*. Springer US. URL: https://doi.org/10.1007/978-1-4614-2361-4. doi:10.1007/978-1-4614-2361-4.

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

Poli, R., Langdon, W. B., & McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd.

Rodrigues, N. M., Batista, J. E., & Silva, S. (2020). Ensemble genetic programming. In T. Hu, N. Lourenço, E. Medvet, & F. Divina (Eds.), *Genetic Programming* (pp. 151–166). Cham: Springer International Publishing.

Đurasević, M., & Jakobović, D. (2020). Automatic design of dispatching rules for static scheduling conditions. *Neural Computing and Applications*, *33*, 5043–5068. URL: https://doi.org/10.1007/s00521-020-05292-w. doi:10.1007/s00521-020-05292-w.

Vázquez-Rodríguez, J. A., & Petrovic, S. (2009). A new dispatching rule based genetic algorithm for the multi-objective job shop problem. *Journal of Heuristics*, *16*, 771–793. URL: https://doi.org/10.1007/s10732-009-9120-8. doi:10.1007/s10732-009-9120-8.

Virgolin, M. (2021). Genetic programming is naturally suited to evolve bagging ensembles. In *Proceedings of the Genetic and Evolutionary Computation Conference* GECCO '21 (p. 830–839). New York, NY, USA: Association for Computing Machinery. URL: https://doi.org/10.1145/3449639.3459278. doi:10.1145/3449639.3459278.

Vlašić, I., Đurasević, M., & Jakobović, D. (2019). Improving genetic algorithm performance by population initialisation with dispatching

rules. *Computers & Industrial Engineering*, *137*, 106030. URL: https://www.sciencedirect.com/science/article/pii/S0360835219304899. doi:https://doi.org/10.1016/j.cie.2019.106030.

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

Wang, S., Mei, Y., & Zhang, M. (2019b). Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 1093–1101).

Wu, L., & Wang, S. (2018). Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks. *International Journal of Production Economics*, *201*, 26–40. URL: https://www.sciencedirect.com/science/article/pii/S0925527318301683. doi:https://doi.org/10.1016/j.ijpe.2018.04.013.

Xu, B., Mei, Y., Wang, Y., Ji, Z., & Zhang, M. (2021). Genetic programming with delayed routing for multiobjective dynamic flexible job shop scheduling. *Evolutionary Computation*, *29*, 75–105. URL: https://doi.org/10.1162/evco_a_00273. doi:10.1162/evco_a_00273.

Yu, L., Shih, H. M., Pfund, M., Carlyle, W. M., & Fowler, J. W. (2002). *IIE Transactions*, *34*, 921–931. URL: https://doi.org/10.1023/a:1016185412209. doi:10.1023/a:1016185412209.

Zahmani, M. H., & Atmani, B. (2019). A data mining based dispatching rules selection system for the job shop scheduling problem. *Journal of Advanced Manufacturing Systems*, *18*, 35–56. URL: https://doi.org/10.1142/S0219686719500021. doi:10.1142/S0219686719500021. arXiv:https://doi.org/10.1142/S0219686719500021.

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

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2021b). Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling. *IEEE Transactions on Cybernetics*, (pp. 1–15). doi:10.1109/TCYB.2021.3050141.

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2021c). Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *25*, 552–566. doi:10.1109/TEVC.2021.3056143.

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

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

Zhang, F., Mei, Y., & Zhang, M. (2019). A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference* GECCO '19 (p. 347–355). New York, NY, USA: Association for Computing Machinery. URL: https://doi.org/10.1145/3321707.3321790. doi:10.1145/3321707.3321790.

Zhang, F., Nguyen, S., Mei, Y., & Zhang, M. (2021f). Multitask learning in hyper-heuristic domain with dynamic production scheduling. In *Ge-*

netic Programming for Production Scheduling (pp. 249–269). Springer Singapore. URL: https://doi.org/10.1007/978-981-16-4859-5_13. doi:10.1007/978-981-16-4859-5_13.

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

Đumić, M., & Jakobović, D. (2021). Ensembles of priority rules for resource constrained project scheduling problem. *Applied Soft Computing*, *110*, 107606. URL: https://www.sciencedirect.com/science/article/pii/S1568494621005275. doi:https://doi.org/10.1016/j.asoc.2021.107606.

Đumić, M., Šišejković, D., Čorić, R., & Jakobović, D. (2018). Evolving priority rules for resource constrained project scheduling problem with genetic programming. *Future Generation Computer Systems*, *86*, 211 – 221. URL: http://www.sciencedirect.com/science/article/pii/S0167739X1732441X. doi:https://doi.org/10.1016/j.future.2018.04.029.

Đurasević, M., & Jakobović, D. (2017a). Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. *Genetic Programming and Evolvable Machines*, *19*, 53–92. URL: https://doi.org/10.1007/s10710-017-9302-3. doi:10.1007/s10710-017-9302-3.

Đurasević, M., & Jakobović, D. (2017b). Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. *Genetic Programming and Evolvable Machines*, *19*, 9–51. URL: https://doi.org/10.1007/s10710-017-9310-3. doi:10.1007/s10710-017-9310-3.

Đurasević, M., & Jakobović, D. (2019). Creating dispatching rules by simple ensemble combination. *Journal of Heuristics*, *25*,

959–1013. URL: https://doi.org/10.1007/s10732-019-09416-x. doi:10.1007/s10732-019-09416-x.

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

Đurasević, M., & Jakobović, D. (2022). Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics. *Journal of Computational Science*, (p. 101649). URL: https://www.sciencedirect.com/science/article/pii/S1877750322000667. doi:https://doi.org/10.1016/j.jocs.2022.101649.

Đurasević, M., Jakobović, D., & Knežević, K. (2016). Adaptive scheduling on unrelated machines with genetic programming. *Applied Soft Computing*, *48*, 419–430. URL: https://www.sciencedirect.com/science/article/pii/S1568494616303519. doi:https://doi.org/10.1016/j.asoc.2016.07.025.

Đurasević, M., Planinić, L., Gala, F. J. G., & Jakobović, D. (2022). Novel ensemble collaboration method for dynamic scheduling problems. URL: https://arxiv.org/abs/2203.14290. doi:10.48550/ARXIV.2203.14290 accepted for publication at GECCO 2022.