Noname manuscript No.
(will be inserted by the editor)

Creating Dispatching Rules by Simple Ensemble Combination

Marko Đurasević · Domagoj Jakobović

Received: date / Accepted: date

Abstract Dispatching rules are often the method of choice for solving scheduling problems since they are fast, simple, and adaptive approaches. In recent years genetic programming has increasingly been used to automatically create dispatching rules for various scheduling problems. Since genetic programming is a stochastic approach, it needs to be executed several times to ascertain that good dispatching rules were obtained. This paper analyses whether combining several dispatching rules into an ensemble leads to performance improvements over the individual dispatching rules. Two methods for creating ensembles of dispatching rules, based on the sum and vote methods applied in machine learning, are used and their effectiveness is analysed with regards to the size of the ensemble, the genetic programming method used to generate the dispatching rules, the size of the evolved dispatching rules, and the method used for creating the ensembles. The results demonstrate that the generated ensembles achieve significant improvements over individual automatically generated dispatching rules.

Keywords Genetic programming · Dispatching rules · Unrelated machines environment · Ensemble learning · Scheduling

1 Introduction

Scheduling is a form of a decision making process in which a set of activities is allocated to limited resources over a certain period of time [46]. Scheduling problems can be found in many real life domains, such as scheduling in air traffic control [7], [14]; semiconductor manufacturing [45]; and therapy scheduling in hospitals [44]. Different metaheuristic methods, including genetic algo-

M. Đurasević · D. Jakobović
Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia
E-mail: marko.durasevic@fer.hr
E-mail: domagoj.jakobovic@fer.hr

rithms and ant colony optimisation, can be used to solve various scheduling problems. These methods, if not coupled with some other approaches (like the *rolling horizon* approach), can be applied only in static scheduling environments. Therefore, simple heuristics in the form of dispatching rules (DRs) are often used for solving dynamic scheduling problems, since they can create schedules in time which is negligible when compared to the execution time of most metaheuristic algorithms [50].

To eliminate the need of manually creating DRs, genetic programming (GP) [47], [25] and similar procedures are often used to generate them automatically. GP has demonstrated the ability to generate DRs of good quality for various scheduling problems, which can even outperform manually designed DRs in many cases. However, since GP is a stochastic procedure which can get stuck in local optima and thus produce bad results, it is advisable that the entire procedure is executed several times to increase the probability of obtaining a good DR. Therefore, several DRs are usually generated, but only one of them will be selected to create schedules for new problem instances.

The objective of this paper is to analyse the creation of ensembles of DRs by simple ensemble combination (SEC). This approach has already demonstrated to obtain significantly better results than several more sophisticated ensemble learning approaches [51]. Unfortunately, in that study the approach was briefly analysed, without providing much insight into how different elements can influence the performance of SEC. Therefore, this paper analyses how three different factors influence the performance of the generated ensembles. The first factor is the influence of creating ensembles by using DRs generated by different GP variant like standard GP, dimensionally aware GP, and gene expression programming. The second factor is the influence of the size of the evolved DRs on the ensemble performance. Finally, different ways of selecting the DRs which will form the ensemble is also analysed. To gain further insights into the created ensembles, this paper provides an analysis of the composition of the best constructed ensembles.

The remainder of this paper is organised as follows. Section 2 gives a literature overview concerned with automatic design of DRs by using GP. Section 3 gives an overview of the unrelated machines scheduling environment, while Section 4 describes the procedure for creating DRs by using GP, as well as how several DRs are combined to form an ensemble and perform scheduling decisions. The results are presented in Section 5. Discussion about the results is given in Section 6, while a short conclusion and steps for further research are given in Section 7.

2 State of the art

Since GP can evolve quite complex expressions and rules, it has extensively been applied in the field of hyper-heuristics [6], [5]. GP has often been used for automatic creation of new DRs. Dimopoulos and Zalzala [8] were among the first to use GP to evolve DRs for the single machine environment, and demon-

strated that the evolved DRs outperform some standard manually designed DRs. Miyashita [30] used GP to evolve DRs for the job shop environment. In his work, Miyashita considered the scheduling problem as a multi-agent problem, where a single agent denoted one machine. Based on that consideration, he proposed three models: the homogeneous model, the distinct agent model, and the mixed agent model. In the homogeneous model the same DR was evolved for each machine. On the other hand, the distinct agent model evolves a different DR for each machine. Finally, the mixed agent model combines the two aforementioned models so that it develops two types of DRs. One DR would be assigned to machines which represent bottleneck resources, while the second DR would be assigned to all other machines. Although the mixed agent model achieved the best results, it requires prior knowledge about the scheduling environment to determine which machines represent bottleneck resources. Apart from evolving DRs for the single machine and job-shop environments, Jakobović *et al.* [20] have also proposed a GP method which extends the mixed agent model of Miyashita. In their approach they generate three expressions, two of which represent DRs, while the third represents a decision function. The decision function is used to determine if a specific machine represents a bottleneck resource or not, and depending on the result one DR will be used if the machine represents a bottleneck, and the other if it does not represent a bottleneck resource. As a consequence, this approach can be used in dynamic environments, since no prior knowledge about the scheduling environment is required.

Tay and Ho [49] used GP to design DRs for a multi-objective scheduling problem. They converted the multi-objective optimisation problem into a single-objective problem by linearly combining all the objectives. Hildebrandt et al. [16] have performed an extensive analysis of creating DRs for the job-shop environment, especially concerning the creation of problem instances. Gene expression programming (GEP) [12], a procedure with similar characteristics as GP, has also been used to evolve DRs for the single-machine [40] and job-shop [39] environments. More recently a new adapted GEP representation was proposed to evolve DRs for the job shop scheduling problem, which significantly improves the results of GEP [52]. Jakobović and Marasović [21] have further investigated scheduling problems in the single-machine and job-shop environments. The focus of their work was not only to analyse the influence of GP parameters on the quality of the obtained DRs, but also to evolve DRs for the single-machine environment which included additional constraints like set-up times and precedence constraints. Different solution representations have been analysed by Nguyen et al. [33], and it was shown that the quality of DRs depends heavily on the representation which is used to design new DRs. Nguyen et al. [35] proposed a novel procedure for evolving iterative dispatching rules (IDRs). This procedure, although applicable only in the static scheduling environment, can create schedules with much better performances when compared to standard DRs. Đurasević et al. [10] compared several GP approaches for creating DRs in the unrelated machines environment, including GEP, IDRs, and dimensionally aware GP. The problem of global perspective of DRs has

been analysed in [17]. In [18] GP was used to evolve DRs for the static two machine job-shop environment for which an optimal algorithm exists. The aforementioned paper demonstrated that GP evolved DRs which can create optimal schedules. GP has also been used to evolve DRs for the order acceptance and scheduling (OAS) problem [42], [31], [37]. The study demonstrated that GP can evolve DRs which create better schedules than those obtained by some standard heuristics designed for the OAS problem.

Nguyen et al. have analysed the possibility of creating entire scheduling procedures (SPs), which consist of both DRs and due-date assignment rules (DDARs) [32], [36]. Both expressions have been evolved simultaneously by using a multi-objective cooperative coevolution GP procedure, which obtains better results in comparison to some traditional multi-objective GP methods. These papers also demonstrated that the generated SPs can outperform SPs created as combinations of different popular DRs and DDARs. Evolving DRs for multi-objective scheduling problems was analysed by Nguyen et al. in [34] and [38]. In those studies a multi-objective problem was considered and several different multi-objective algorithms were used to create DRs for optimising five criteria simultaneously. Karunakaran et al. [22] used two multi-objective algorithms to evolve DRs which optimise three criteria simultaneously. Đurasević and Jakobović [11] use four multi-objective algorithms to optimise various multi-objective scheduling problems. Recent papers also started to investigate feature selection to determine which features are relevant for creating new DRs [19], [29]. GP was recently used to generating DRs for scheduling problems which were not considered until now, like the dual-constrained flow shop scheduling problem [3] and the intercell scheduling problem [27]. A recent survey conducted by Branke et al. gives a detailed overview of the literature concerned with automatic creation of DRs by the use of GP [4].

Ensemble learning techniques have of yet rarely been used for creating ensembles of dispatching rules. Park et al. [43] proposed an ensemble learning GP procedure which uses cooperative coevolution to evolve ensembles of DRs. It was demonstrated that the ensemble learning GP procedure generally produced more robust rules than the single rule GP. Unfortunately, this approach was applied only for the static scheduling problem. Hart and Sim [15] proposed a new hyper-heuristic ensemble learning method named NELLI-GP. This method creates the ensemble in a way that each DR contained in the ensemble is adjusted for optimising only a certain subset of training instances. Therefore, the ensemble will consist of DRs where each rule will focus on solving a different subset of problem instances. Although this approach achieves good results, it was also applied only on the static job shop scheduling problem. In [41], Park et al. compare several combination schemes for creating ensembles for the dynamic job shop environment. The authors also perform a detailed analysis to investigate the behaviour of the ensemble when using each of the tested combination schemes. Đurasević and Jakobović [51] made a comparison of several ensemble learning methods on the dynamic unrelated machines scheduling problem. The aforementioned paper shows that SEC achieved better results than more sophisticated ensemble learning methods like BagGP,

BoostGP, and cooperative coevolution. Unfortunately, SEC was not analysed extensively in the paper. Therefore, the objective of this paper is to further study SEC to improve its performance and analyse the composition of ensembles it creates. The main contributions of this paper, when compared to the one where the method is proposed, are: proposing several methods for constructing ensembles (concerned especially on reducing the time complexity of the approach), a more detailed analysis of how ensemble size influences the performance, analysing the performance of SEC when applied on DRs generated with different GP variants, and analysing the generated ensembles to obtain a better understanding of the method.

3 Scheduling in the Unrelated Machines Environment

In the unrelated machines environment, a number of n jobs compete to be processed on one of the m available machines. Each job has several attributes which are considered in the scheduling process. For the unrelated machines environment, the following attributes are often included: processing time p_{ij}, which denotes the processing time that is required to process the job with index j on the machine with the index i; release time r_j, which determines the point in time when job j becomes available; due date d_j, which determines a point in time until which the execution of job j should be completed, otherwise a certain cost will be incurred; and weight w_j, which determines the importance of job j.

3.1 Scheduling criteria

Throughout the literature many different scheduling criteria have been defined and optimised [1], [2]. This study will focus on four criteria which have most often been used in the literature.

After each job is scheduled, the following metrics are calculated:

- C_j - completion time of job j
- F_j - flowtime of job j: $F_j = C_j - r_j$.
- T_j - tardiness of job j: $T_j = \max\{C_j - d_j, 0\}$.
- U_j - flag if a job is tardy or not: $U_j = \begin{cases} 1 : T_j > 0 \\ 0 : T_j = 0 \end{cases}$.

Based on the previously defined metrics, it is possible define the following four scheduling criteria which will be optimised in this paper:

- C_{max} - maximum completion time of all jobs: $C_{max} = \max_j\{C_j\}$,
- Twt - total weighted tardiness: $Twt = \sum_j w_j T_j$,
- Ft - total flowtime: $Ft = \sum_j F_j$,
- Nwt - weighted number of tardy jobs: $Nwt = \sum_j w_j U_j$.

3.2 Dynamic scheduling

In this paper it is presumed that scheduling is performed under dynamic conditions. This means that the properties of the jobs are not known in advance, rather they become known at the moment when the job enters the system (which is determined by the release times of jobs). DRs are more than appropriate for creating schedules in such environments, due to their speed and since they only consider the currently available jobs when performing the decision. They can be used in parallel with the execution of the system, and each time a job is released into the system they can immediately schedule it on the next available machine. Therefore, DRs can quickly react to the changing conditions which can occur in the scheduling environment.

4 Methodology

This section will describe the procedure which was used for automatic creation of new DRs, and the procedure of combining DRs to create ensembles of DRs.

4.1 Automatic development of DRs

The DR used in this paper consists of two independent parts: a meta-algorithm and a priority function. The meta-algorithm is used to schedule each job on a certain machine. It does so by employing a certain priority function which calculates the priority value for each given job and machine. The meta-algorithm used in this study is defined manually, and its pseudo-code is given in Algorithm 1. Each time a job is released into the system or a machine becomes idle the meta-algorithm is triggered. By using the priority function the meta-algorithm calculates priorities for job-machine pairs constructed from all released but unscheduled jobs and all machines (idle or not). For each job the meta-algorithm then determines the machine for which the lowest priority value is obtained. If the selected machine is free, the job is scheduled on it. On the other hand if the machine is not free, the job is not scheduled, and the meta-algorithm will try to schedule the job at a later moment in time, with the possibility of scheduling it on another machine. In this way the meta-algorithm will not necessarily schedule a job on a machine immediately when it becomes idle, but rather it can leave the machine free if it determines that it would be better to schedule the jobs on machines that are currently busy. Thus, it is important that the priority values for a job are calculated for all machines, whether they are idle or not. It is important to outline that the meta-algorithm does not depend on any concrete priority function, which means that many different priority functions can be used with the given meta-algorithm. As a consequence the same meta-algorithm can use priority functions which optimise different scheduling objectives.

On the other hand, priority functions are automatically generated by using GP. Since priority functions are defined as mathematical functions, one of

Algorithm 1 Meta-algorithm used for GP scheduling

```
while unscheduled jobs are available do
    wait until a job becomes ready or a job finishes;
    for all released unscheduled jobs and all machines do
        obtain the priority π_ij of job j on machine i;
    end for
    for all available jobs do
        determine the best machine (the one for which
        the best value of priority π_ij is achieved);
    end for
    while jobs whose best machine is available exist
     do
        determine the best priority of all such jobs;
        schedule the job with the best priority;
    end while
end while
```

Table 1: Terminal nodes

Node name	**Description**
pt	processing time of job j on the machine i (p_{ij})
pmin	The minimal job processing time on all machines: $\min\{p_{ij}\}\forall i$
pavg	The average processing time on all machines
PAT	Patience - the amount of time until the machine with the minimal processing time for the current job will be available
MR	Machine ready - the amount of time until the current machine becomes available
age	The time that the job spent in the system: $time - r_j$
Terminals used only for the total weighted tardiness and number of tardy jobs criteria	
dd	Due date (d_j)
SL	Positive slack: $max\{d_j - p_{ij} - time, 0\}$
w	Weight

the most important steps is to define the building blocks that will be used to construct them. In the context of GP these building blocks can be grouped either as terminal or function nodes. Terminal nodes represent system attributes that are used by GP for creating DRs. The list of used terminal nodes is shown in Table 1. This set was obtained by performing extensive experiments and selecting those terminals which provide useful information. Some nodes were specifically defined for the unrelated machines environment (PAT and MR), while others were inspired by some standard DRs (SL). Aside from the terminal nodes, a set function nodes is also defined. These nodes represent mathematical operators used to combine terminal nodes into meaningful expressions. In this study five different operators are used for the set of function nodes: addition operator, subtraction operator, multiplication operator, protected division operator $/(a,b) = \begin{cases} 1, & \text{if } |b| < 0.000001 \\ \frac{a}{b}, & \text{else} \end{cases}$, and the POS operator $POS(a) = max\{a, 0\}$.

Apart from the standard GP procedure, two other models for generating DRs will also be used: GEP and dimensionally aware GP (DAGP) [24]. These

methods were included to determine if the procedure used for creating priority functions can have a significant influence on the quality of the ensemble. GEP is a procedure which is similar to GP, but unlike in GP the evolved expression is not represented in the form of a tree, but rather by using an array of constant size [12]. Because of this, it is easier to implement and design genetic operators. On the other hand, DAGP is a procedure which embeds semantic information into the expressions evolved by GP. This is achieved in a way that each node in the expression contains additional semantic information (in most cases a unit) and that several semantic rules, which determine how nodes with different semantic information can be combined, are defined. With these two elements it is possible to introduce units in the expression and ensure that the expression does not perform illegal operations on units (like adding seconds to meters). The advantage of such a procedure is that it can evolve solutions which are easier to interpret and adhere to all given semantic rules. In our implementation the initialisation method and genetic operators were adapted so that all individuals satisfy the semantic rules. However, it is also possible to allow that certain individuals violate the given rules and that the degree by which they are violated is reduced by using methods like the culling method [24] or the penalty method [28].

The parameters used by the described GP procedures are listed in Table 2. If a parameter is not applicable for a specific GP approach, the value in the cell is denoted with "-". In cases where more than one genetic operator is defined, the algorithm will randomly select one of the listed operators in each iteration (all operators have the same probability of being selected). Since it would be too expensive to perform the parameter optimisation for each criterion which is considered in this paper, the parameter values were optimised and fine-tuned for the Twt criterion, and the obtained values were applied when creating DRs for the other criteria as well. The parameters were optimised individually by testing several different values. The population size was tested with values of 200, 500, 1000, 2000; the termination criteria was tested with values up to 240 000 iterations; the mutation probability was tested with values 0.07, 0.1, 0.2, 0.3, 0.5; tree depths of maximum sizes equalling to 3, 5, 7, 11, and 13 were tested; different crossover and mutation operator combinations were also tested (the operator combinations were generated by simple heuristics which tried to improve the fitness of the GP by adding or removing certain operators). GEP was tested with head sizes of 6, 8, 10, 12, and 14 nodes, while values of 2, 3, 4, and 5 were tested for the number of genes. For each value the GP algorithm was executed 50 times and the best DR from each run was saved. The average of the fitness values achieved by the best individuals from the 50 runs was calculated, and the parameter value which achieved the smallest average value was selected. By performing the parameter optimisation procedure it was possible to improve the average value of the generated DRs by around 3%, but also to reduce the standard deviation of the obtained solution by more than a half of the value. However, since the parameter optimisation procedure is a quite time consuming and lengthy process, it would be desirable that this step could be avoided. Therefore, it is inter-

Table 2: Parameters for the GP procedures

Parameter	GP	DAGP	GEP
Population size	1000		
Stop criteria	maximum number of iterations (80 000)		
Selection	steady state GP using tournament selection		
Tournament size	3		
Initialization	*ramped half-and-half*		random
Mutation probability	0.3		
Maximal tree depth	5		-
Crossover operators	subtree, uniform, context-preserving, size-fair	one point	
Mutation operators	subtree, Gauss, hoist, node complement, node replacement, permutation, shrink	node replacement	
Head size	-		6
Gene count	-		3
Transposition operators	-		IS, RIS, gene transpostion

esting to analyse how the SEC method would perform if it were to use DRs evolved by GP with the initial unoptimised parameters. For that reason, SEC will also be applied on DRs obtained by the unoptimised GP procedure to analyse if parameter optimisation has a significant influence on the ensemble quality. The parameter values for the unoptimised GP (UGP) procedure are the same as those shown in Table 2, except for the maximum tree depth which was equal to 7, and the set of crossover operators which additionally included the one-point crossover operator. The values for UGP were chosen as a rule of thumb based on the experience from certain preliminary experiments, prior to the parameter optimisation.

4.2 Simple ensemble combination of DRs

Although the previously described GP approach can produce new DRs which outperforms standard manually designed DRs, there is always a need to improve the performance of automatically designed DRs. Unfortunately, achieving significant improvements in the quality of generated DRs is not trivial. For example, by increasing the number of iterations GP will usually start to overfit on the training set, which will result in DRs that generalise poorly on unseen problem instances. Therefore, GP is usually augmented with different approaches to improve its performance. Nevertheless, even with such improved GP it is hard to design a single DR which performs well in all situations. Since GP is a stochastic optimisation procedure it also needs to be executed several times to increase the probability of generating a DR of good quality. However, usually only one of the evolved DRs is used for scheduling, while the other DRs are discarded although they could contain good characteristics and perform well in certain situations.

By creating ensembles of DRs it is possible to deal with both of the aforementioned problems to a certain degree. The idea of SEC is to use a set of evolved DRs and combine them into ensembles which should perform better than individual DRs by themselves. The ensembles are expected to achieve

better results since they perform the decision based on all the individual DRs in the ensemble. This reduces the probability of performing a suboptimal decision in a certain situation, since it is unlikely that all DRs will perform a poor decision. As a consequence, the ensembles are also expected to be more stable than individual DRs, since their decision is performed based on several DRs. The reason why this ensemble method is considered in this paper instead some other ensemble learning methods (like bagging or boosting) is due to several reasons. First of all, the benefit of this approach is that previously generated DRs can be used, thus reducing the computational effort in comparison to other methods. This means that ensembles can be constructed extremely fast if previously generated DRs exist. Even if there is a need to generate new DRs for ensemble construction, it can be done in a way that each DR is generated independently, similarly as in bagging but without the need to sample the problem instances in each iteration. In addition, the SEC method is simple in the fact that it does not introduce additional complexity in the construction of the individual DRs, unlike bagging and boosting which require that either the problem instances are sampled for each DR or that the influence of individual instances is adapted. Although the SEC method is quite simple, it demonstrated to perform equally well and even better than other traditional ensemble learning methods in many cases.

The rest of this section will explain the two key parts of the SEC approach: how the decisions of all DRs in the ensemble are combined and how the DRs forming the ensemble are selected.

4.2.1 Combination of DRs

One of the most important things that need to be specified for SEC is the way in which the ensemble performs its decision based on the DRs which are contained in the ensemble. To combine the decisions of all DRs into a single decision two combination methods, based on similar procedures from the machine learning field [48], are used: sum and vote. The meta-algorithm denoted in Algorithm 1 uses the results obtained from these combination methods to determine which job should be scheduled on which machine.

The *sum combination method* is defined as an ensemble rule which sums the priority values of all DRs contained in the ensemble. The result of that sum represents the priority value which will be assigned to a job-machine pair. This value is then used by the meta-algorithm to schedule jobs on machines in a way that the job-machine pair which received the lowest priority value will be selected for scheduling. The obvious advantage of this approach is its inherent simplicity. However, there is one open issue with this method that needs to be addressed and explained. The issue is that if the ensemble consists of DRs which are generated independently, then there is no guarantee that the priority values obtained from the different DRs will be of the same order of magnitude. Therefore, it is possible that some DRs in the ensemble would not have any effect on the decisions of the ensemble, since the contribution of their priority values would be insignificant when compared to the priority values

of other DRs. This issue could be solved by normalising the priority values obtained by each DR. However, this can not be easily done since the range of priority values which can be obtained by a DR is not known in dynamic environments. In static environments it would be possible to determine the range of priority values for each DR, for example by using *interval arithmetic* [23]. However, this issue does not influence the ensemble decisions as much as it would be expected, since it was shown that the priority values of different DRs are usually of the same order of magnitude, and thus each DR in the ensemble will have a similar influence on the decision of the ensemble.

In the *vote combination method*, denoted in Algorithm 2, each DR casts a vote for the element that received the lowest priority value by that rule. The element which received the most votes will be selected for scheduling. A problem with this procedure is that ties can appear more often than with the sum combination method. To alleviate this problem, voting is not performed for all machines and jobs immediately, but will be split up in several parts. In the first part, for the currently considered job each DR will cast a vote for the machine that received the best priority value for that job. The machine which received the most votes is assigned to the considered job. If this was the first job for which a machine was assigned to, the procedure is repeated for the next unscheduled job which is already released into the system. However, if there already exists job-machine pair, then the vote method is used again to determine which of those two pairs is better. In this second step each DR casts a vote for either job-machine pair, and the one which receives the most votes is kept, while the other is deleted. This procedure is repeated for all unscheduled, but released jobs. In the end of the procedure, one job-machine pair will remain, and if the machine in that pair is free, the job which was associated to the machine will be scheduled on it. If two pairs receive the same number of votes, then the one whose job has the earlier release time will be selected. Naturally, a better way could have been chosen to deal with ties, for example by using the sum method, but for this study it was decided to focus on the two simple methods, and to leave the possibilities of their improvements for further research. An advantage of this method over the sum combination method is that the magnitudes of priorities are not important, since each DR casts a vote of the same weight.

4.2.2 Creation of ensembles of DRs

The second important step of SEC is to define how the ensembles of DRs are created. For that purpose it is mandatory to define the size of the ensemble, and the procedure for selecting DRs that should form the ensemble.

The size of the ensemble represents an important parameter since both the quality of the ensemble and the time needed to perform the scheduling decision will depend on it. Additionally, the execution time of the methods used for creating the ensembles also depends on the size of the ensemble which needs to be generated. Therefore, smaller ensembles are preferred to larger ones. Fortunately, this does not affect the performance of the SEC approach, since

Algorithm 2 The vote combination method

```
Let bestPair represent the best selected job-machine pair (empty at the beginning)
for each unscheduled job which is already released into the system do
    for each DR in the ensemble do
        Calculate the priority value by using the selected DR for all machines
        Determine the machine for which the DR achieved the best value and vote for it
    end for
    Select the machine with the most votes
    Let currentPair denote the job-machine pair chosen in this iteration
    if bestPair is not empty then
        for each DR in the ensemble do
            Make a vote between currentPair and bestPair
        end for
        if currentPair received more votes than bestPair then
            bestPair ← currentPair
        end if
    else
        bestPair ← currentPair
    end if
end for
Schedule the job in the bestPair on the machine in the bestPair
```

in a previous study it was demonstrated that the SEC approach achieves the best results for the ensemble size of five DRs [51]. The influence of different ensemble sizes will also be analysed in this paper as well, to gain a better understanding of its influence on the results.

The most difficult part of SEC is to define a procedure for choosing which DRs will form the ensemble. Since scheduling is performed in dynamic conditions the ensemble needs to be constructed in advance. Five methods will be used for constructing ensembles: random selection method, probabilistic selection method, grow method, grow-destroy method, and instance based method.

The simplest of the applied ensemble construction methods is the *random selection method*. This method constructs the ensemble in a way that it randomly selects which DRs should form the ensemble. In this method all DRs have the same probability of being selected for the ensemble. Unfortunately, the probability of obtaining a good ensemble by trying out only one combination of DRs is quite small. Therefore, instead of constructing a single ensemble, this method constructs a number of ensembles, evaluates them on the training set, and selects the one which achieved the best value for the optimised objective. The number of ensembles which will be created and tested is a parameter of this approach. The complexity and execution time of this approach will increase as the value of the parameter increases. The disadvantage of this approach is that the ensemble construction is performed completely randomly, without using any information about the quality of the ensembles or DRs.

An extension of the previous approach is the *probabilistic selection method*. In this method each DR has a different probability of being selected, which depends on the quality of DRs on the training set. Therefore, better DRs will have a higher probability of being selected into the ensemble. In the tested

method the probability of an individual was defined to be proportional to the value of its fitness. The motivation behind this method is to guide the search towards those DRs which have a better individual fitness, in hope that better ensembles will consist out of DRs which individually also perform better.

The *grow method* is a greedy heuristic which builds the ensemble incrementally. The method starts with a single DR, which can be selected randomly by its fitness value, or by some other criteria. In each iteration of the method, the DR which increases the quality of the ensemble the most is added to the ensemble. This is done until the ensemble reaches its specified size. Unlike the previous two construction methods, this one behaves deterministically given that the same initial DR is used. The motivation behind this method is that it should choose those DRs which increase the overall quality of the ensemble.

The *grow-destroy method* represents an extension of the grow method. The first part of the method is the same as in the grow method, however, instead of building up an ensemble of the specified size e, this method creates an ensemble twice the size. After this step is done, the method performs e iterations, where in each iteration the method removes the DR whose removal would result in the best fitness of the smaller ensemble. As the previous method, this one is also deterministic given the same starting DR. The motivation for this method is to allow for the ensemble to grow over the specified size and allow it to collect more good DRs in the ensemble, but then in the next step to remove those ensembles which least contribute to the quality of the ensemble.

Finally, the *instance based method* behaves similarly as the grow method. However, it does not use the fitness of the ensembles or individuals, but rather the number of problem instances on which the ensemble achieves the best results out of all available DRs. In the first step for each problem instance the minimum objective value, which any of the available DRs achieve, is determined. The first DR of the ensemble can be chosen randomly, by its fitness, or some other criteria. The method then iteratively constructs the ensemble by adding DRs which achieve the best performance on those problem instances on which the ensemble does not obtain a result which is at least equally good as the best result obtained for that problem instance. This approach does not necessarily guide the search towards ensembles which have a better overall fitness, but rather to those ensembles which perform well on as many problem instances as possible. Thus, such a selection method should produce an ensemble which would be suitable for solving a variety of problem instances, and produce good results on unseen problem instances. Another benefit of this approach is that at each iteration it evaluates only one ensemble, since DRs are selected based on their individual performance, and only the ensemble at the end of each iteration needs to be evaluated to determine on which problem instances it does not perform well.

When using the grow, grow-destroy, and instance based methods it is possible that after an insertion of a DR in the ensemble its fitness would deteriorate in comparison with the smaller ensemble size. In this study such situations are ignored and the procedures continue to add new rules until the specified size

is reached. In future research it would be interesting to adapt the approaches in a way that it also optimises the size of the ensemble.

5 Results

5.1 Benchmark Setup and Evaluation

The effectiveness of the proposed methods will be evaluated using scheduling problem instances generated based on some methods described in related references [26], [9], [20], [13]. In total 120 problem instances were generated, from which 60 instances are used for the training set, and 60 instances for the test set. Depending on the problem instance, the number of jobs can be 12, 25, 50, or 100, while the number of machines can be 3, 6, or 10. The total fitness value of an individual is calculated as a sum of objective values for each individual problem instance. More details about the problem instance generation procedure can be found in the Appendix.

To test the performance of SEC 50 DRs are first evolved by GP using the training set. The SEC method is then applied on these 50 generated DRs to construct ensembles of DRs using the same training set. The ensemble is constructed by using one of the afore described ensemble construction methods. Since the random selection and probabilistic selection methods are stochastic, both of those methods were executed 30 times to determine if the results obtained by those two methods are statistically better than those of the individual DRs. The other three construction methods are not stochastic given a concrete starting DR. Since he performance of the ensemble still depends the initially selected DR, the method is executed 50 times, each time a using a different initial DR. The performance of the generated ensembles is evaluated on the test set. When comparing the results between different SEC variants the median value of the generated ensembles will be used. On the other hand, the comparison between the ensembles and the individual DRs used to construct them will be performed differently. Out of the DRs that were generated by GP the one which performs the best on the training set will be selected and its performance will be compared with the median value of the ensembles on the test set. The intuition behind such a comparison is that if several DRs are already evolved by GP then it should be better to select the one which achieves the best performance on the training set since it should have a greater chance of performing well on the test set rather than if a random DR is selected. However, an alternative comparison would be that only a single DRs is generated by GP and compared to the constructed ensembles. In this way usually a slightly worse result is obtained then by using the best DR on the training set, but it is not required to construct as many DRs and thus the computational effort can be significantly reduced.

Additionally, the Mann-Whitney statistical test is used to determine if there is a significant difference between the results obtained between different methods. The results are considered statistically significant if the obtained p

value is smaller than 0.05. Aside from the results for the generated ensembles, the tables will also include the result of the DR which achieved the best performance on the training set out of the evolved DRs. This rule will be denoted as *Best DR* in the tables and figures. Furthermore, the best ensemble values for each approach will be denoted in bold, while the results of ensembles in which the median value is better than the result of the best DR on the training set are denoted with a grey cell.

5.2 Results for creating ensembles with different ensemble construction methods

In this section the five proposed methods for the construction of ensembles will be compared. All the ensemble construction approaches will be tested only on the Twt criterion, since presenting results for all the criteria would be too extensive. However, results for the other criteria are included in the Appendix. The DRs used for creating the ensembles were created by using the standard GP approach. The random and probabilistic selection methods have been tested for different values of the number of ensembles they will create in one run. The number of ensembles they create will be additionally denoted beside the name of the construction methods in the tables and figures.

Table 3 represents the results achieved by various ensemble construction methods for the sum combination method, while Figure 1 represents those results using a box-plot representation. The first thing which can be noticed is that in all cases the ensembles perform better on the average than the best individual DRs, and that in some cases the improvement over the individual DR ranged up to 6%. A further benefit is that most of the ensembles generated by SEC achieve a better performance than the individual DRs, which can be seen from the box plots. Except for a few outlier solutions, most of the results are better than the best DRs. The best improvements over the standard DRs are usually achieved by ensembles consisting of either five or seven DRs. Although all the ensemble construction methods achieved better results, the quality of the ensembles depends both on the size of the constructed ensemble and the construction method. For the smallest ensemble size the best results are obtained by the random and probabilistic methods. However, for larger ensemble sizes the instance based method obtained the best results.

The random selection method performs the best when using ensembles of sizes 5 and 7. For the ensemble size of 3 the best results are obtained when creating a larger number of ensembles. The overall best result is obtained for the ensemble size of 7 and when constructing 500 ensembles, which is significantly better than any other result obtained by the SEC method. A similar behaviour can also be observed for the probabilistic selection method. Based on all the results it can be concluded that both of the aforementioned methods perform very similarly, since neither achieves better results consistently. Out of the other three selection methods the instance based method consistently obtained the best results, especially as the ensemble size increases. This

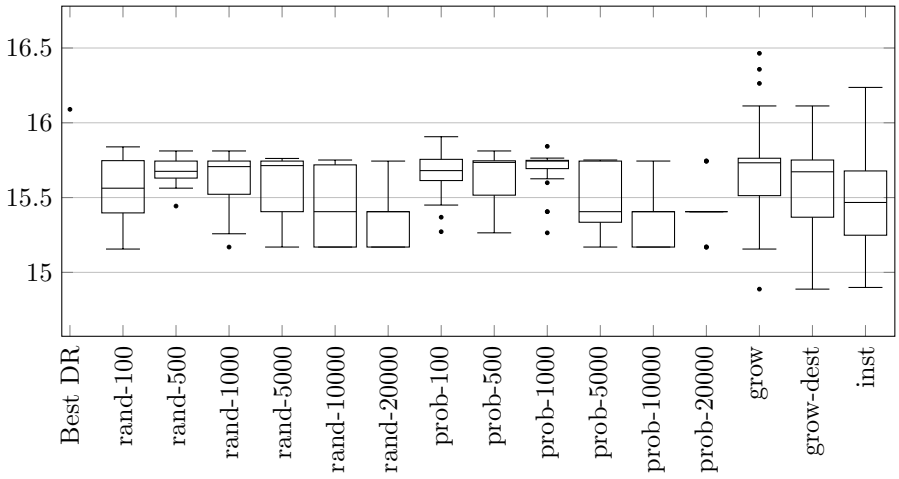
Table 3: Comparison of the ensemble construction approaches for the sum combination method

Procedure	Ensemble size								
	3			5			7		
	min	med	std	min	med	std	min	med	std
Best DR		16.09			16.09			16.09	
Rand 100	15.16	15.58	0.203	15.03	15.38	0.199	15.01	15.37	0.194
Rand 500	15.44	15.68	**0.086**	15.00	15.52	0.226	14.98	15.29	**0.144**
Rand 1000	15.17	15.71	0.185	15.22	15.61	**0.172**	15.09	15.40	0.186
Rand 5000	15.17	15.71	0.234	15.22	15.60	0.243	15.08	15.43	0.228
Rand 10000	15.17	**15.41**	0.244	15.21	15.50	0.243	15.10	15.38	0.244
Rand 20000	15.17	**15.41**	0.162	15.05	15.52	0.197	15.06	15.40	0.242
Prob 100	15.27	15.69	0.129	15.12	15.37	0.218	14.96	15.34	0.204
Prob 500	15.26	15.74	0.172	15.15	15.61	0.206	15.02	15.40	0.227
Prob 1000	15.26	15.74	0.124	15.13	15.57	0.197	15.10	15.43	0.198
Prob 5000	15.17	15.56	0.231	15.18	15.47	0.197	15.05	15.41	0.189
Prob 10000	15.17	**15.41**	0.197	15.15	15.40	0.241	15.00	15.51	0.242
Prob 20000	15.17	**15.41**	0.132	15.14	15.44	0.286	15.17	15.47	0.251
Grow	**14.89**	15.73	0.290	15.06	15.52	0.299	15.12	15.30	0.288
Grow-dest	**14.89**	15.67	0.230	15.10	15.30	0.247	15.10	15.44	0.254
Inst	14.90	15.47	0.324	**14.98**	**15.28**	0.307	**14.84**	**15.09**	0.350

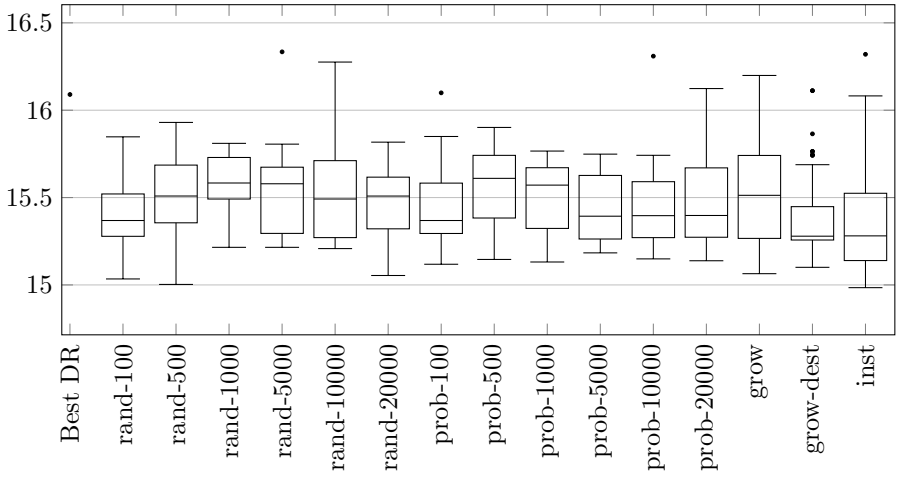
method has obtained the best result, which is significantly better than the results obtained by any of the other methods.

Table 4 shows the results achieved for different ensemble construction methods when using the vote combination method. Figure 2 shows a box-plot representation of the results. When the results are compared to the standard DRs it is evident that except for four experiments the ensembles perform better on average than the best DR. Similarly as with the vote combination method it can be seen that most of the constructed ensembles achieve a better performance than the best DR. The improvements of ensembles over the individual DR reach up to 4%. The best improvement is achieved by the probabilistic selection method when creating 10000 ensembles and for the ensemble size of three DRs. However, there is no statistical difference between this result and the results obtained by most of the other ensemble construction methods, which is due to the fact that most methods achieved quite similar results. The random selection method achieves the best result for the ensemble size of 7, being only slightly worse than the probabilistic selection method. However, the statistical tests show that there is no significant difference between these two methods. The other three methods achieve a similar performance as the aforementioned two methods.

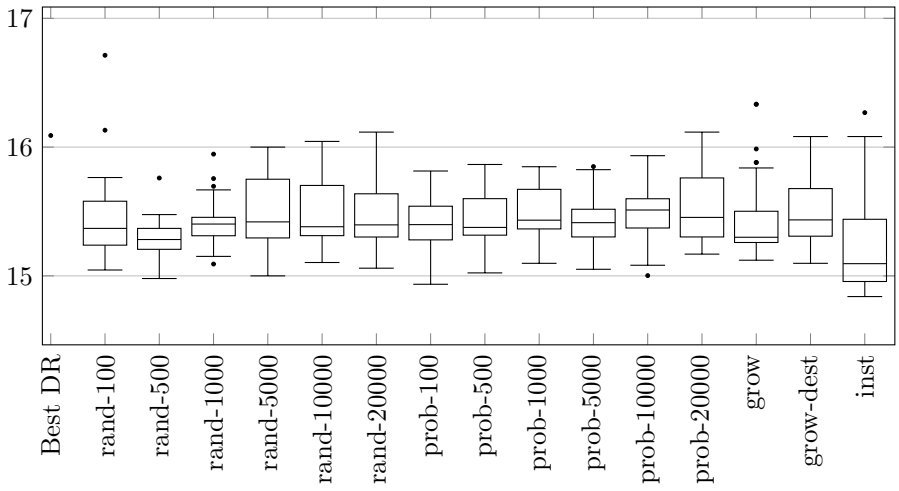
By comparing the vote and sum combination methods several conclusions can be drawn. For the smallest ensemble size it is evident that both combinations methods achieve a very similar performance. As the ensemble size increases the sum combination method outperforms the vote combination method for most of the tested GP methods. Statistical results show that in most cases the sum method was significantly better than the vote combina-



(a) Comparison of the ensemble construction methods for ensembles of size three

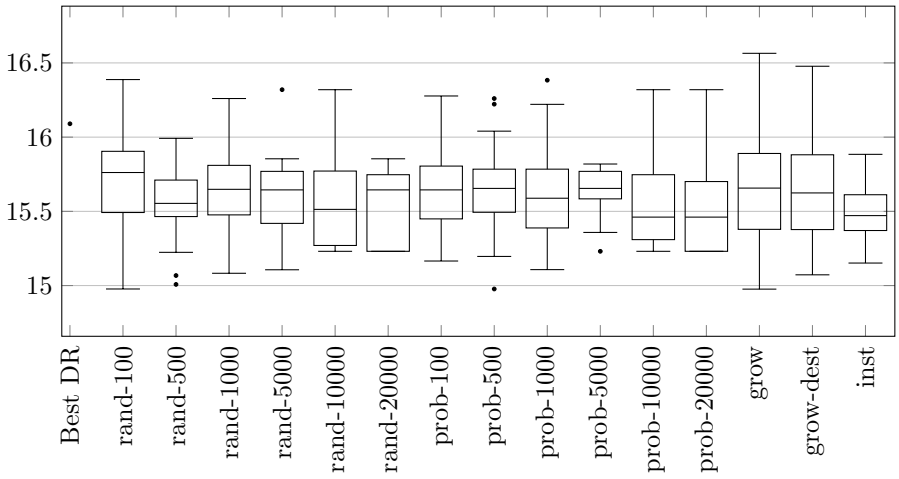


(b) Comparison of the ensemble construction methods for ensembles of size five

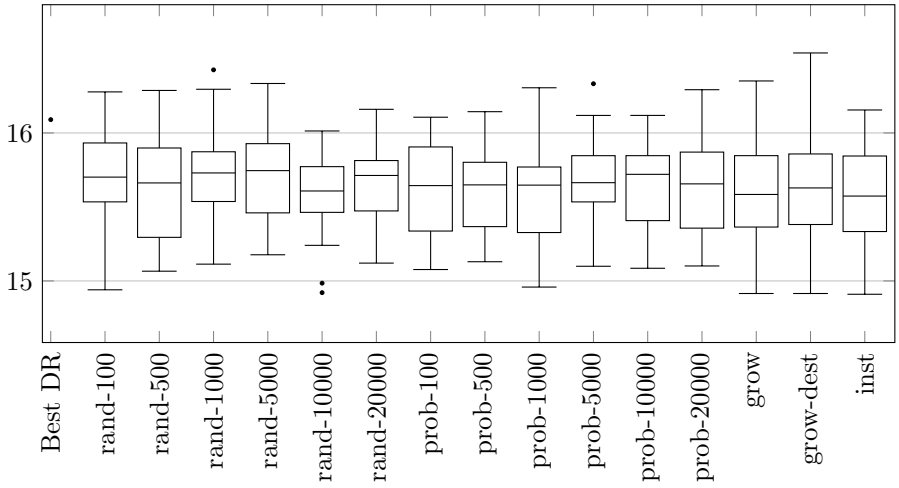


(c) Comparison of the ensemble construction methods for ensembles of size seven

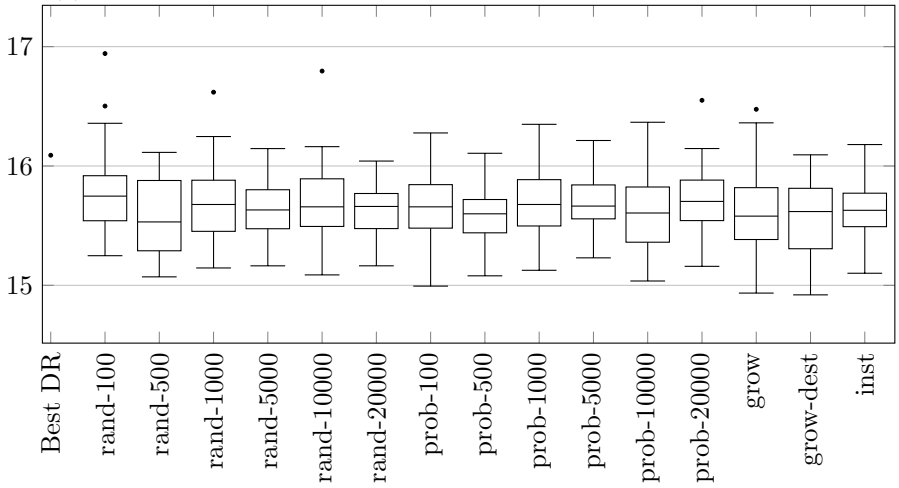
Fig. 1: Box-plot representation of results for the ensemble construction approaches and the sum combination method



(a) Comparison of the ensemble construction methods for ensembles of size three



(b) Comparison of the ensemble construction methods for ensembles of size five



(c) Comparison of the ensemble construction methods for ensembles of size seven

Fig. 2: Box-plot representation of results for the ensemble construction approaches and the vote combination method

Table 4: Comparison of the ensemble construction approaches for the vote combination method

Procedure	Ensemble size								
	3			5			7		
	min	med	std	min	med	std	min	med	std
Best DR		16.09			16.09			16.09	
Rand 100	**14.98**	15.76	0.341	14.94	15.71	0.294	15.25	15.76	0.347
Rand 500	15.01	15.60	0.238	15.07	15.67	0.348	15.07	**15.55**	0.330
Rand 1000	15.08	15.65	0.295	15.11	15.73	0.310	15.15	15.68	0.341
Rand 5000	15.11	15.65	0.246	15.18	15.76	0.328	15.16	15.65	0.236
Rand 10000	15.23	15.57	0.280	14.92	15.62	**0.266**	15.09	15.67	0.342
Rand 20000	15.23	15.64	0.248	15.12	15.71	0.233	15.16	15.67	**0.216**
Prob 100	15.17	15.65	0.282	15.08	15.66	0.312	14.99	15.66	0.319
Prob 500	**14.98**	15.66	0.278	15.13	15.66	0.279	15.08	15.60	0.242
Prob 1000	15.11	15.60	0.320	14.96	15.65	0.320	15.13	15.68	0.301
Prob 5000	15.23	15.68	**0.149**	15.10	15.69	0.282	15.23	15.61	0.330
Prob 10000	15.23	**15.46**	0.300	15.09	15.75	0.272	15.04	15.61	0.330
Prob 20000	15.23	15.55	0.271	15.10	15.66	0.313	15.16	15.71	0.282
Grow	**14.98**	15.66	0.374	14.92	15.61	0.329	14.93	15.62	0.332
Grow-dest	15.07	15.63	0.361	14.92	15.63	0.340	**14.92**	15.62	0.289
Inst	15.15	15.47	0.161	**14.91**	**15.57**	0.330	15.10	15.65	0.223

tion method. This just shows that the performance of the sum combination method improves with the size of the ensemble, while on the other hand the performance of the vote combination method was mostly constant.

Based on the results denoted in this section it is difficult to select the most appropriate ensemble construction method. Since the random and probabilistic selection methods achieve a similar performance, out of those two the random selection method was selected for further experiments, due to the fact that it is simpler. As for the number of ensembles which the method creates, the method usually performs the best when creating a smaller number of ensembles. Thus it was decided that the random selection method will create 500 ensembles and select the best one. Out of the other three methods the instance based method clearly outperformed most of the other methods. Although the instance based method obtained better results than the random selection method in the previous tests, it was still decided to use the random selection method in subsequent experiments. The reason for this is due to the fact that additional experiments demonstrated that the instance based method can be unstable in certain situations. Therefore, in the rest of this paper the random selection method with 500 constructed ensembles will be used.

5.3 Results for different ensemble sizes

This section will analyse the performance of SEC when constructing ensembles of different sizes. Table 5 represents the obtained results, while Figure 3 shows the box-plot representation of those results. The table includes results up to the ensemble size of 20 since the results started to stagnate for the larger sizes. The

first thing which can be noticed is that for almost all experiments the ensemble construction method obtained better average results than the best DR. In most cases all of the generated ensembles achieved a better performance than the best DR. This shows that regardless of the ensemble size the generated ensembles can outperform the best DR. For smaller ensemble sizes, especially with the vote combination method, the ensembles are only slightly better than the best DR, however, as the ensemble size increases so does the performance of the ensembles improve.

When using the sum combination method the results improve quite fast until a size of around seven DRs is reached, after which the improvements in the results are minor, and there is no significant improvement in the results with the increase of the ensemble size. The best median value is obtained when using an ensemble size of 15. After this ensemble size it is evident that the performance of the method slowly deteriorates with the increase of the ensemble size. Thus it seems that for the sum combination method the best results are obtained when using ensemble sizes between 7 and 19. For the vote combination method the performance does not improve as fast as for the sum combination method. Furthermore, the improvements that are obtained for the larger ensemble sizes are not significantly better than those obtained for some smaller sizes. When comparing the results obtained by the two combination methods, it is evident that the sum combination method obtains better results. Although both methods perform similarly for smaller ensemble sizes, as the ensemble size increases the sum combination method obtains significantly better results than the vote combination method.

In the end both construction methods obtained the best results for ensembles of size 15. However, statistical tests demonstrated that there is no significant improvement in the results obtained by this ensemble size and several smaller ensemble size. Thus it was decided to use the smallest ensemble size for which both combination methods do not achieve significantly worse results than ensembles of size 15. The smallest ensemble size for which this condition holds is the size 7. In addition to this ensemble size it was also decided to also use two smaller ensemble sizes, those of 3 and 5, to get a better understanding how the method performs for different ensemble sizes across different situations. The reason why the paper is focused more on smaller ensembles is due to the fact that they can be constructed and evaluated faster. Fast evaluation is especially important if the ensembles are used under dynamic conditions. Furthermore, smaller ensembles are more easier to interpret and analyse, which can be important for better understanding them. However, the results demonstrate that using larger sizes should not have a negative effect on the performance until a certain ensemble size.

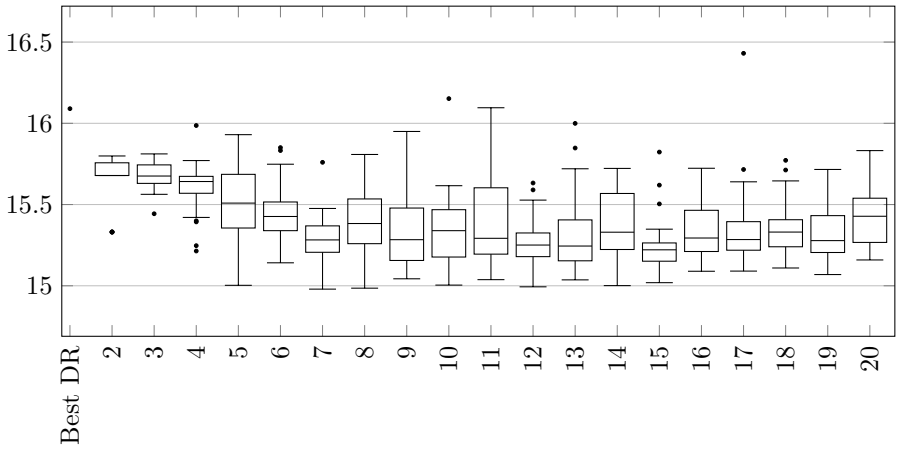
Table 5: The results of the random selection method when constructing ensembles of different sizes

	Sum			Vote		
	min	med	std	min	med	std
best DR		16.09			16.09	
2	15.33	15.68	0.137	15.13	15.76	0.321
3	15.44	15.68	0.086	15.01	15.60	0.238
4	15.21	15.65	0.154	14.97	15.67	0.367
5	15.00	15.52	0.226	15.07	15.67	0.348
6	15.14	15.43	0.175	15.30	15.64	0.270
7	**14.98**	15.29	0.144	15.07	15.55	0.330
8	**14.99**	15.38	0.201	15.06	15.57	0.240
9	15.04	15.32	0.258	15.01	15.52	**0.185**
10	15.00	15.34	0.220	15.01	15.55	0.276
11	15.04	15.30	0.289	**14.99**	15.58	0.261
12	**14.99**	15.25	**0.147**	15.11	15.68	0.265
13	15.04	15.25	0.249	15.30	15.68	0.220
14	15.00	15.34	0.205	15.14	15.53	0.238
15	15.02	**15.22**	0.166	15.12	**15.46**	0.220
16	15.09	15.30	0.171	15.15	15.68	0.255
17	15.09	15.29	0.254	15.29	15.66	0.255
18	15.11	15.33	0.176	15.25	16.68	0.226
19	15.07	15.28	0.173	15.16	15.57	0.210
20	15.16	15.44	0.181	15.26	15.54	0.201

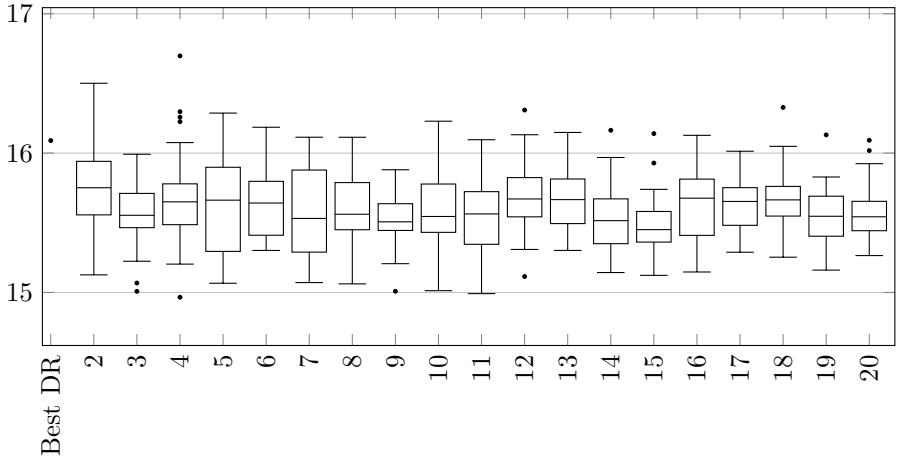
5.4 Results for constructing ensembles of DRs evolved by different GP variants

This section will analyse the influence of using different GP variants (GP, UGP, GEP, and DAGP) for generating DRs on the quality of ensembles created from these DRs. Table 6 represents the results achieved by SEC for the different GP variants, while Figure 4 shows the box-plot representation of the results.

The first thing which can be noticed from the results is that for most experiments SEC achieves better results on average than the best DR. Out of the two combination methods, the sum method displayed some problems in certain situations and did not achieve better result than the best DR on average. On the other hand, the vote combination method obtained better results on average than the best DR for all experiments. These results demonstrate that the vote combination is more stable, and thus achieves better results more consistently. Across the DRs generated by different GP methods SEC obtained the best improvements for the Twt and Nwt criteria, which can reach up to 5.6%. For the other two criteria the method did not achieve an equally good improvement in the results, but it still obtained better results on average than the best DR. The SEC method also obtains a good distribution of results compared to the best DRs, which can be seen from the fact that most of the generated ensembles achieve solutions better than the best DR. This is



(a) Comparison different ensemble sizes for the sum combination method



(b) Comparison different ensemble sizes for the vote combination method

Fig. 3: Box-plot representation of results for different ensemble sizes

especially true for the vote combination method which usually lead to better distributed solutions.

On all four tested GP variants SEC achieved a similar improvement in the results, which shows that the method used for generating DRs does not have a large effect on the performance of ensembles. The figures show that in many experiments the ensembles obtained by SEC achieve better performance than the best DR. Out of the tested GP variants, SEC achieved the best median values for the Twt and Nwt criteria when using DRs generated by DAGP, while for the other two criteria the best median values were obtained by DRs generated by GEP. It is interesting that when using UGP SEC did not fare worse than when using optimised GP, since for the Nwt and C_{max} criteria there

Table 6: Results for constructing ensembles of DRs evolved by different GP approaches

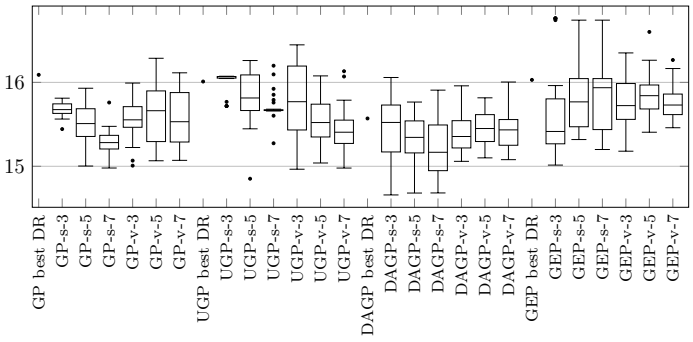
Approach			Twt			Nwt			Ft			Cmax		
	Ensemble Type	Size	min	med	std	min	med	std	min	med	std	min	med	std
GP	Best DR			16.09			8.192			159.4			38.68	
	sum	3	15.44	15.68	0.203	8.129	8.333	0.129	158.5	159.7	**0.357**	38.41	38.74	0.178
		5	15.00	15.52	0.226	7.824	8.144	0.154	158.0	159.1	0.577	38.50	38.94	0.130
		7	**14.98**	**15.29**	**0.144**	**7.634**	8.175	0.275	157.9	159.1	0.456	38.46	38.94	0.149
	vote	3	15.01	15.60	0.238	7.665	7.941	0.162	157.8	158.8	0.584	38.36	38.54	0.121
		5	15.07	15.67	0.348	7.759	**7.931**	0.116	157.8	**158.7**	0.437	**38.35**	**38.52**	**0.113**
		7	15.07	15.55	0.330	7.708	7.943	**0.100**	**157.7**	**158.7**	0.503	**38.35**	38.53	0.115
UGP	Best DR			16.01			8.380			161.0			38.75	
	sum	3	15.72	16.07	**0.116**	7.773	8.154	0.239	159.2	159.9	0.430	38.50	38.74	0.070
		5	**14.85**	15.82	0.290	7.450	7.954	0.273	159.1	159.7	**0.360**	38.47	38.59	0.100
		7	15.27	15.67	0.156	7.843	8.181	**0.130**	158.7	159.4	0.488	38.47	**38.55**	**0.055**
	vote	3	14.98	15.82	0.434	7.644	7.871	0.136	**158.2**	159.2	0.612	**38.34**	38.67	0.188
		5	15.04	15.52	0.244	7.637	**7.810**	0.172	158.3	159.1	0.387	38.38	38.65	0.122
		7	14.98	**15.41**	0.279	**7.405**	7.822	0.186	**158.2**	**159.0**	0.435	38.49	38.60	0.096
DAGP	Best DR			15.57			8.061			159.4			38.69	
	sum	3	**14.66**	15.53	0.360	7.473	7.765	0.127	157.6	158.5	0.541	38.48	38.60	0.110
		5	14.68	15.34	0.300	**7.467**	**7.694**	0.127	157.5	158.7	0.584	38.47	38.63	0.107
		7	14.68	**15.19**	0.325	7.531	7.818	0.147	158.2	158.9	0.495	38.45	**38.55**	0.097
	vote	3	15.06	15.36	0.216	7.581	7.938	0.170	**157.4**	158.7	0.527	38.42	38.58	**0.071**
		5	15.10	15.46	**0.191**	7.726	7.955	0.114	157.6	**158.4**	0.512	**38.41**	38.59	0.097
		7	15.08	15.45	0.237	7.780	8.004	**0.110**	**157.4**	158.5	**0.473**	38.43	38.60	0.082
GEP	Best DR			16.03			7.991			159.5			38.49	
	sum	3	**15.01**	**15.43**	0.547	7.682	8.107	0.125	157.6	158.5	0.547	38.40	38.52	0.079
		5	15.32	15.77	0.361	7.750	7.832	0.239	157.6	158.5	**0.484**	38.46	38.53	**0.049**
		7	15.2	15.93	0.444	**7.613**	**7.735**	0.129	157.6	**158.3**	0.639	38.41	38.55	0.061
	vote	3	15.18	15.73	0.281	7.698	7.875	0.123	**157.3**	**158.3**	0.512	38.40	**38.47**	0.052
		5	15.41	15.85	0.256	7.705	7.882	0.170	157.6	158.8	0.612	38.40	**38.47**	0.055
		7	15.46	15.73	**0.200**	7.687	7.900	**0.114**	157.6	159.1	0.557	**38.38**	38.49	0.080

was no significant difference, while for the other two criteria SEC obtained better results when using DRs generated by the optimised GP variant. This demonstrates that parameter optimisation is not of great importance when using SEC, since it can still construct ensembles of good quality.

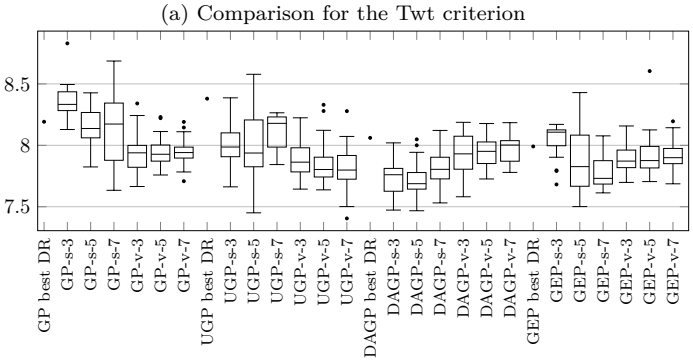
5.5 Results for creating ensembles from DRs of different sizes

This section presents the results achieved by SEC when using DRs generated by the standard GP with different maximum tree depths. The motivation behind the tests in this section is to analyse how the size of individual DRs influences the quality of the entire ensemble. For that purpose DRs created with maximum tree depths of three, five, and seven will be tested. This means that the expressions generated by using the depth value of three can have a maximum number of 15 nodes, by using the depth value of five a maximum number of 63 nodes, and by using the depth value of seven a maximum number of 255 nodes. The GP method which uses the maximum tree depth of three is denoted as GP3, the GP which uses the maximum tree depth of five as GP5, and the GP which uses the maximum tree depth of seven as GP7.

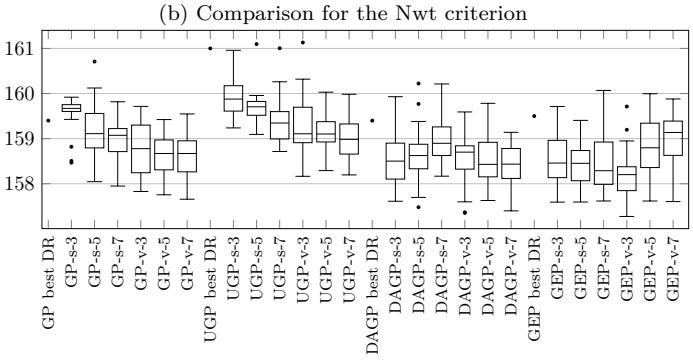
Table 7 represents the results achieved by the ensembles constructed from DRs of different maximum sizes, while Figure 5 shows the box-plot represen-



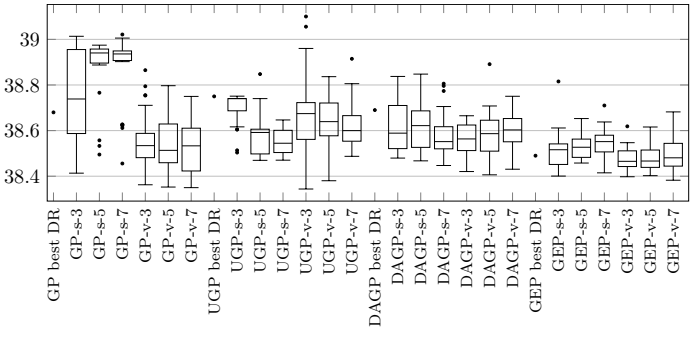
(a) Comparison for the Twt criterion



(b) Comparison for the Nwt criterion



(c) Comparison for the Ft criterion



(d) Comparison for the Cmax criterion

Fig. 4: Box plot representation of results for constructing ensembles of DRs evolved by different GP approaches

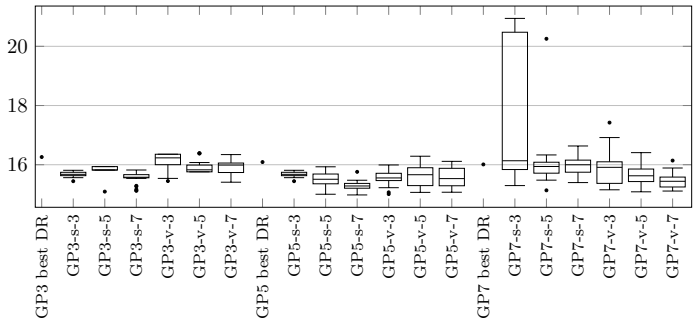
Table 7: Results for constructing ensembles of DRs of different sizes

Approach			Twt			Nwt			Ft			Cmax		
	Ensemble Type	Size	min	med	std	min	med	std	min	med	std	min	med	std
GP3	Best DR			16.26			8.010			160.2			38.68	
	sum	3	15.25	**15.45**	0.260	7.921	7.950	0.092	**158.7**	159.4	0.661	38.48	38.69	0.108
		5	**15.09**	15.82	**0.152**	7.835	**7.956**	**0.071**	**158.7**	159.1	0.418	**38.37**	38.62	0.137
		7	15.12	15.58	0.184	7.876	7.982	0.135	**158.7**	**159.0**	**0.188**	38.46	38.55	0.108
	vote	3	15.45	16.23	0.249	7.873	8.047	0.128	159.1	159.9	0.406	38.45	**38.51**	**0.105**
		5	15.76	15.83	0.192	**7.765**	7.990	0.131	158.9	159.6	0.414	38.42	38.70	0.116
		7	15.41	15.99	0.235	**7.765**	7.990	0.131	159.0	159.4	0.290	38.42	38.69	0.110
GP	Best DR			16.09			8.192			159.4			38.68	
	sum	3	15.44	15.68	0.203	8.129	8.333	0.129	158.5	159.7	**0.357**	38.41	38.74	0.178
		5	15.00	15.52	0.226	7.824	8.144	0.154	158.0	159.1	0.577	38.50	38.94	0.130
		7	**14.98**	**15.29**	**0.144**	**7.634**	8.175	0.275	157.9	159.1	0.456	38.46	38.94	0.149
	vote	3	15.01	15.60	0.238	7.665	7.941	0.162	157.8	158.8	0.584	38.36	38.54	0.121
		5	15.07	15.67	0.348	7.759	**7.931**	0.116	157.8	**158.7**	0.437	**38.35**	**38.52**	**0.113**
		7	15.07	15.55	0.330	7.708	7.943	**0.100**	**157.7**	**158.7**	0.503	**38.35**	38.53	0.115
GP7	Best DR			16.01			8.134			158.7			39.34	
	sum	3	15.30	16.15	2.214	7.858	8.067	0.129	158.7	159.7	0.850	38.41	38.54	0.211
		5	15.13	15.95	0.841	7.917	8.129	**0.081**	158.6	159.4	0.823	38.40	38.86	0.206
		7	15.40	16.00	0.286	7.698	8.162	0.181	**158.4**	159.6	0.897	38.39	38.88	0.185
	vote	3	15.15	15.91	0.529	**7.476**	7.795	0.169	158.5	**159.1**	0.319	38.33	**38.49**	0.208
		5	**15.08**	15.63	0.329	7.531	7.804	0.140	158.6	159.3	0.399	**38.30**	38.69	0.162
		7	15.11	**15.45**	**0.249**	7.624	**7.775**	0.122	158.7	**159.1**	**0.312**	38.44	38.63	**0.111**

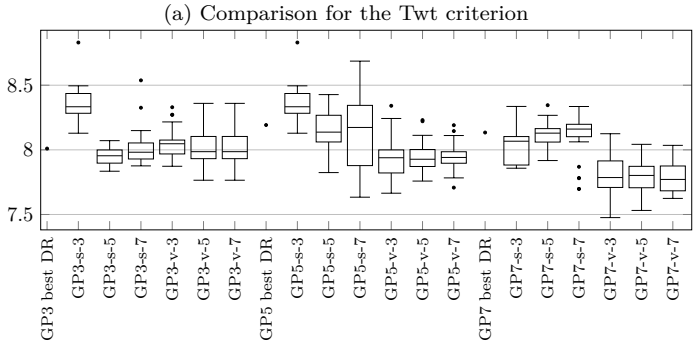
tation of the results. The table shows that in most experiments the ensembles obtained better results on average than the best DR. The performance the sum combination method was again quite volatile since in many cases, especially for the larger DR sizes, SEC did not obtain better results than the best DR. With the vote combination method SEC achieved better results on average than the best DR in almost all experiments. For the smallest DR sizes, generated by GP3, the sum combination method usually obtained better median values of the results. However, for larger DR sizes the vote combination method clearly achieved better results. As for the performance of the ensembles, the best median values for all criteria, except Nwt, were obtained when using DRs generated by the maximum tree depth of 5. For Nwt the best results were obtained by using the largest tree depth. Therefore the tree depth of 5 seems to be appropriate for constructing ensembles, since it offers the best results across most of the criteria.

5.6 Results for different DR collections

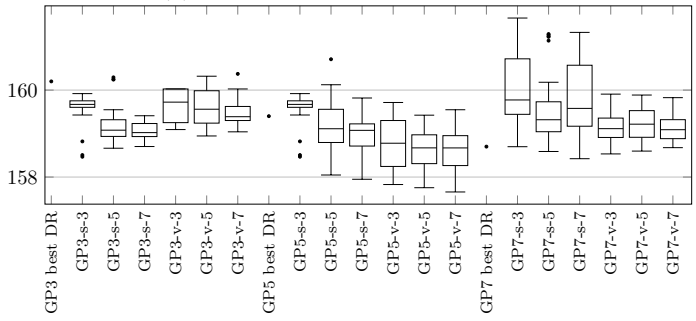
This section will test how different DR collections generated by GP influence the performance of the SEC method. To test this, GP was used to generate 30 sets each consisting out of 50 DRs. For each of these sets the best DR on the training set is determined and then executed on the test set to evaluate its performance. After that the SEC method is executed on each of these sets to generate 30 ensembles for each set. For each set the median value of the 30 generated ensembles is calculated and collected. The 30 values of the selected individual DRs are then compared with the 30 median values which were



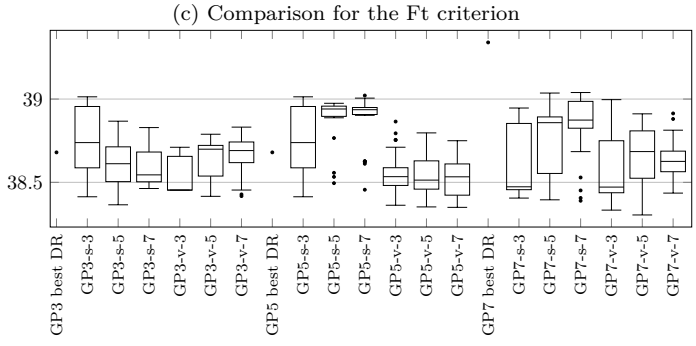
(a) Comparison for the Twt criterion



(b) Comparison for the Nwt criterion



(c) Comparison for the Ft criterion



(d) Comparison for the Cmax criterion

Fig. 5: Comparison between ensembles of DRs of different maximum sizes

Table 8: Performance of SEC on different sets of DRs

Method		min	med	std
Individual DRs		15.27	16.02	0.887
	Ensemble			
Combination	Size	min	med	max
sum	3	15.26	15.65	0.313
	5	15.34	15.84	0.273
	7	15.19	15.84	0.343
vote	3	15.23	15.67	0.165
	5	15.44	15.60	0.111
	7	15.36	15.53	0.134

obtained when generating ensembles. In that way it is possible to determine whether generating ensembles with SEC provides a benefit over generating many DRs and selecting the one which performs best on the training set. Since these tests require a significant amount of GP runs to be performed, the analysis was performed only for the Twt criterion.

Table 8 demonstrates the performance of the individual DRs and ensembles constructed on 30 different DR sets. The table shows that for all the tested parameters the ensembles obtained a better median value than the best individual DRs. The statistical tests demonstrate that the SEC method obtained significantly better results than the best individual DRs for all the tested parameters. This is even better evident from the box-plot in Figure 6, where it is evident that the results of the ensembles are much less dispersed that those of the individual DRs. Thus the ensembles are more likely to to produce better results than it would be the case when the DR which achieved the best performance on the test set would be used. Out of the two combination methods the vote method performed better since it obtained a smaller median value and less dispersed results. The obtained results demonstrate that even over a large number of DR collections the SEC method constructs ensembles which on average perform significantly better than the best DR on the training set would be selected from the available DRs. This justifies using SEC to construct ensembles rather than simply selecting the best DR on the training set.

6 Discussion

6.1 Influence of ensemble construction methods on the ensemble performance

The ensemble construction method represents one of the most important parts of SEC, therefore it is expected to have a large influence on the performance of the constructed ensembles. The results presented in the last section show that there is indeed some difference between the construction methods in their performance, however, no single method achieves the overall best performance and can be considered superior to the others. Apart from the quality of the

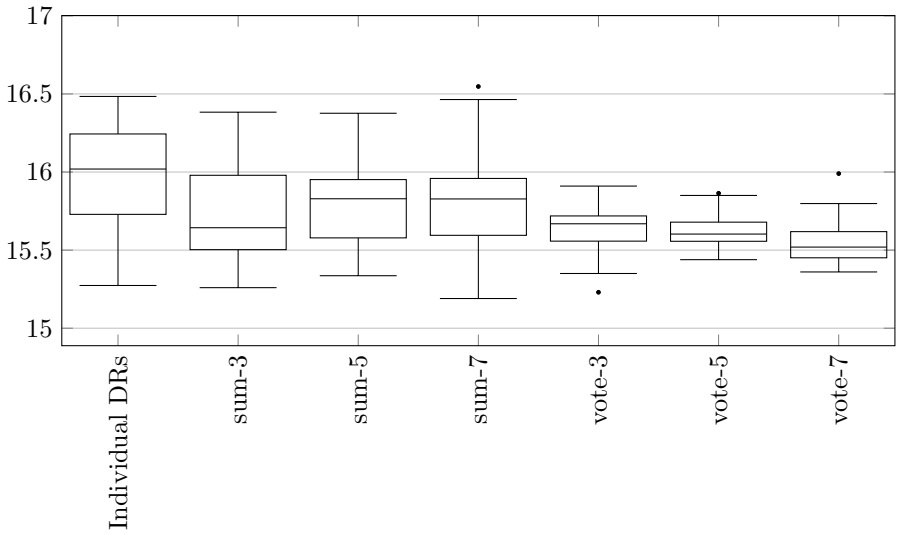


Fig. 6: Performance of SEC on different sets of DRs

evolved ensembles, different construction methods also differ in the number of ensembles they create during their execution.

Random selection is the simplest of the tested methods, which obtained overall good results. In the focus of this method was the analysis of the number of ensembles which it constructs in each run. Although it might seem that the method should perform best when a larger number of ensembles is constructed, the results show that this is not necessarily the case. For the smallest ensemble size (that of 3) the method obtained better results when a larger number of ensembles was constructed. However, with larger ensemble sizes the best results were usually obtained when either 100 or 500 ensembles were created. The reason for such behaviour is due to the fact that larger ensembles are more adaptable, and thus if many combinations are tried out it is likely that the method will choose an ensemble that is more appropriate for solving the instances in the training set, rather than having a good generalisation property. Therefore, this method should be used with a smaller number of constructed ensembles, usually of a few hundreds. This is an important conclusion since it shows that without a significant loss in performance the random combination method can create a smaller number of ensembles than it was suggested in the previous study. The probabilistic selection method was defined in an attempt to improve the performance of the random selection method by giving a greater selection probability to DRs with better individual quality. However, this method usually delivered results which were similar to the random selection method. Thus it seems as if the individual quality of DRs does not provide relevant information for constructing ensembles. Since the obtained results show no improvement, but the method is more complex due

to calculation of the probabilities for the DRs, it is evident that the random selection method can be regarded as a better choice.

The motivation for the other three proposed methods is to guide the search towards good ensembles, but instead of using the individual fitness, the influence of the selected DR on the entire ensemble is used. For most experiments all three methods achieved results that were close to those obtained by the random selection method that constructs a smaller number of ensembles. The grow and grow-destroy methods achieved in most cases similar results and thus neither of these two methods can be considered as superior. On the other hand the instance based method achieved the best results out of these three methods, achieving significantly better results than the other two methods in several occasions. In some cases the instance based method managed to significantly outperform the random selection method, which demonstrates that it is a powerful construction strategy. However, some experiments revealed several issues with this construction method. The most important drawback of this method is that not only did it not achieve better results for the C_{max} criterion, it usually obtained results that were significantly worse than those of the individual DRs. For the other three criteria the method achieved significantly better results than individual DRs in less occasions than the random selection method, which just supports the fact that the instance based method is less stable. The method achieved results which were sometimes even better than those of the random selection method which means that the selection strategy employed by the instance based method certainly has potential.

The computation complexity of the selection methods can also be an important factor when selecting which one will be used. The complexity of the random and probabilistic selection methods can easily be controlled by the number of ensembles they construct. Since the best results were obtained when creating a smaller number of ensembles, SEC can construct the ensembles in a short amount of time. The grow, grow-destroy, and instance based methods also evaluate a small number of ensembles, which means that they will execute quickly. For example the grow method will evaluate 329 ensembles for the ensemble of size seven, while the grow-destroy method will evaluate 686 ensembles. The instance based method will evaluate even less ensembles, concretely only 6. An additional important thing is that since these methods build the ensembles incrementally, in earlier iterations the ensembles which are constructed are smaller and evaluated faster. Therefore, the execution time of these methods is comparable to the execution time of the random selection method when creating 100 and 500 ensembles, depending on the size of the constructed ensemble. The instance based method can be regarded as the most efficient one, since it does not evaluate the constructed ensembles at all, while the other methods mostly have a similar complexity.

6.2 Influence of GP variants on the ensemble performance

The results obtained by SEC when applied on DRs generated by different GP methods provide some interesting details. First of all it is evident that the method achieves good results on DRs generated by all the different GP variants, especially when the vote combination method was used. This shows that SEC performs well regardless of the method used to generate the DRs. Furthermore, the results show that the maximum improvement that can be obtained by the ensemble over the individual DRs is very similar for the different methods. For example, the maximum improvement for the Twt criterion for the four tested GP variants was between 2.4% and 5%. Therefore, SEC provided a constant improvement in the results over the various GP methods.

An interesting behaviour can be observed when comparing the results obtained when using GP and UGP. It is evident that SEC performed extremely well even when using DRs generated by UGP, for which the parameters were not optimised. Although DRs generated by UGP had a similar median value to GP, the results were much more dispersed, and more outlier values were obtained. However, this did not negatively influence SEC, since it achieved results that were quite similar as the results obtained when using DRs generated by GP. Furthermore, for both DRs generated by both GP variants SEC generated ensembles with similar dispersion, thus the more dispersed DRs of UGP did not have any negative influence on the ensembles. These results show that for the SEC method the parameters of GP do not necessarily need to be thoroughly optimised, and that the SEC method will still obtain good results. This can substantially decrease the time required to obtain results.

When all things are considered, it can be concluded that the SEC method performs well across all DRs generated by different GP variants. Since the differences between the different methods are not large, it can not be concluded that the SEC method would prefer DRs constructed by any GP variant.

6.3 Influence of the maximum DR depth on the ensemble performance

SEC obtained good results for all the tested DR sizes. Therefore, it seems that using larger or smaller DRs does not largely affect the performance of the generated ensembles. As it is evident from the results, the ensembles performed best for most of the criteria when constructed out of DRs with the maximum tree depth of 5. For the smaller and larger DR sizes the ensemble was again able to obtain better results for most of the experiments, but mostly not to such a great extent. Thus, it seems that SEC prefers the use of moderately sized DRs, since too small rules seem too limited, while the larger ones seem to be a bit unstable.

6.4 Influence of the ensemble size and ensemble combination method

The size of the ensemble and ensemble combination method both constitute important parameters that influence the performance of the constructed ensembles. The SEC method was tested with ensemble sizes up to 20 DRs. The results show that both the vote and sum combination method achieved the best median value for the ensemble size 15. However, after a more detailed analysis of the results it was noticed that the results for the larger ensemble sizes did not significantly differ from the results which were obtained by the ensembles of size 7 for both combination methods. Since the smaller ensembles can be constructed and evaluated much faster, the ensemble size of 7 was used. Additionally, smaller ensembles are easier to interpret and analyse. In the tests performed for the different GP variants it was noticed that in several cases the ensembles of smaller sizes obtained significantly better results than larger ensembles. This just demonstrates that for different situations different ensemble sizes are appropriate and that no single ensemble size is the best. The vote method demonstrated to work well with all three tested ensemble sizes, which means that any of them could be used to obtain significantly better results. For the sum combination method the larger ensemble sizes seem to lead to better results.

Out of the two tested ensemble combination methods the vote combination method has shown to be much more stable since in most cases it obtained significantly better results then the sum combination method. On the other hand, the sum combination method was in several occasions unable to achieve better results on average in comparison to those obtained by the best DR. The reason for this is probably due to the fact that it is more difficult to find DRs which interact well for the sum combination method, since different DRs can have various influences in the ensemble. In the vote combination method all DRs have the same influence, since they cast a vote, so it is easier to construct well working ensembles. Nevertheless, SEC constructed several ensembles with the sum combination method that achieved some of the best results. This simply shows that the sum combination method is equally expressive and powerful as the vote combination method. However, since it has clearly shown to be more stable and reliable, the vote combination method should be preferred with SEC.

6.5 Analysis of ensembles

In this section the structure of the ensembles will be analysed with regards to DRs that most often constitute the ensembles. This analysis should provide a deeper insight if there is any regularity in the DRs which are chosen to form the ensembles and if this information could somehow be used in the ensemble construction process. For this purpose the ensembles created for optimising the Twt criterion, using DRs generated by GP, will be analysed. Ensembles of seven will be used for the analysis.

Table 9: Most commonly contained DRs in the best ensembles combined by the sum combination method

Method	DR indices														
	0	1	2	3	4	8	9	10	11	14	15	20	24	33	46
Random	4	8	7	9	5	2	5	10	8	4	17	0	0	3	10
Probabilistic	6	8	14	3	6	4	11	6	5	5	12	1	0	1	8
Grow	48	24	11	5	35	6	19	7	33	4	30	4	3	1	7
Grow-destroy	5	26	11	15	8	27	20	12	19	8	30	0	6	7	9
Instance based	1	1	1	50	1	1	1	1	1	50	9	50	50	50	1
Fitness	14.49	14.49	14.55	14.38	14.76	14.63	14.61	14.63	14.46	14.42	14.40	14.30	14.36	14.39	14.64

6.5.1 Analysis of the frequency of DRs in the ensembles generated by SEC

Table 9 gives an overview of fifteen DRs that most commonly appeared in the best ensemble constructed in each algorithm run with the sum combination method. For the random and probabilistic selection methods 30 runs were performed, while 50 runs were performed for the remaining three methods. Each cell in the table represents the number of times that the DR with the given index appeared in the best constructed ensemble, for a concrete construction method. For each ensemble construction method, the three DRs which most often appeared in the best ensembles are denoted with a grey cell. The fitness values in the tables denote the fitness of the DRs on the training set. The table shows that the different combination methods have a preference for different DRs. DR 15 seems to be the only rule which is often used by all of the combination methods. An interesting thing that the table shows is that the instance based method is biased towards the DRs it will use. This is evident from the fact that it uses 5 DRs in all of the ensembles it constructed, and it shows that the instance based method was very biased towards using DRs with a better individual fitness. The other methods commonly include DRs of a poorer individual quality, which means that they do not necessarily prefer better DRs. Surprisingly, even the probabilistic selection method was not strongly inclined towards using DRs of better individual quality. This just seems to support the fact that DRs which individually perform good do not perform well within an ensemble. A further fact supporting this conclusion is that the best individual DR, the one denoted with the index 20, is seldom included in ensembles constructed by any method except the instance based method.

Table 10 represents the prevalence of DRs in the best ensembles constructed with all the construction methods, and by using the vote combination method. The instance based method is again inclined towards using same DRs in most of the ensembles. The other methods use more distinct DRs to construct good ensembles. The reason for that is most likely due to the fact that a single bad DR in the ensemble will have a less significant influence on the performance of the ensemble when using the vote combination method, than it would be the case with the sum method. A further interesting fact is that DRs which are most commonly selected by the methods when using the vote combination method usually have a worse individual quality than those selected when us-

Table 10: Most commonly contained DRs in the best ensembles combined by the vote combination method

Method	DR indices														
	4	6	7	11	12	15	22	30	32	38	39	40	46	47	48
Random	7	7	3	7	10	3	10	1	8	11	9	2	4	12	7
Probabilistic	3	0	10	10	6	10	5	3	5	8	10	0	6	6	8
Grow	12	8	6	11	13	5	10	11	15	4	14	10	14	12	12
Grow-destroy	8	4	6	11	15	7	7	9	15	7	19	12	9	17	12
Instance based	1	45	1	47	1	50	1	48	4	1	1	1	1	1	1
Fitness	14.76	14.77	14.48	14.46	14.68	14.40	14.51	14.42	14.63	14.87	14.81	14.68	14.64	14.54	14.56

Table 11: Performance of the ensembles constructed out of the most commonly use DRs

	Sum			Vote		
Ensemble size	3	5	7	3	5	7
Random	15.57	15.56	15.34	15.74	15.73	15.48
Probabilistic	15.43	15.45	15.37	15.20	15.53	15.35
Grow	16.49	15.27	15.39	16.58	15.28	15.51
Grow-destroy	15.74	15.27	15.58	15.58	16.20	16.05
Instance based	15.01	15.37	15.10	15.23	16.08	15.84

ing the sum combination method. Therefore the vote combination method is even less inclined towards using good individual DRs to form the ensembles. The combination method has a significant influence on which DRs are most appropriate for creating ensembles, which is evident from the fact that only rules 4, 11, 15, and 46 appear in both tables and thus are most often used by both methods.

One interesting thing that can additionally be tested is to see whether the information about how often the individual DRs are used in ensembles could provide to be useful when constructing new ensembles. To test this assumption the three, five, and seven most commonly used DRs by all the methods were combined into ensembles and evaluated on the test set. The obtained results are presented in Table 11. The obtained results for the sum combination method show that the information about the frequency of DRs can be useful to construct ensembles of good quality. For example, all except one constructed ensemble achieves a better performance than the individual DRs on average. Furthermore, the ensembles tested in this table also often achieved a better or equal performance as the corresponding construction method on average.

6.5.2 Analysis of ensembles generated by SEC

Table 12 represents the performance of the best ensemble of size seven constructed by the random selection method with the sum combination method. The table also denotes the performance of individual DRs which form the ensemble. The analysis will be performed on the test set. Ten instances which seemed to best describe the behaviour of the ensemble on the entire problem set were selected and outlined in the table. The best results of the individual

Table 12: Performance analysis of the best ensemble generated by SEC and combined with the sum combination method

Problem instance index	Individual DR							Ensemble
	1	9	10	12	30	42	49	
3	0.279	0.283	0.268	0.273	0.310	0.849	0.330	0.283
5	0.237	0.237	0.237	0.237	0.246	0.237	0.237	0.168
8	0.537	0.602	0.594	0.516	0.576	0.695	0.523	0.523
17	0.709	0.909	0.733	0.840	1.024	0.858	0.892	0.638
27	0.015	0.100	0.100	0.015	0.015	0.100	0.100	0.015
29	0.266	0.182	0.182	0.182	0.223	0.182	0.223	0.182
33	0.003	0.003	0.003	0.003	0.000	0.003	0.003	0.003
38	0.331	0.364	0.370	0.433	0.418	0.352	0.370	0.354
53	0.915	0.699	0.698	0.698	0.607	0.699	0.708	0.699
57	0.544	0.412	0.452	0.434	0.386	0.418	0.455	0.411
Total fitness on all instances	15.92	15.39	15.50	15.29	15.67	16.47	15.95	14.98
Fitness on the training set	14.49	14.61	14.63	14.68	14.42	14.43	14.81	14.53

DRs for each problem instance are denoted with a grey cell, while the results for which the ensemble performs at least as well as the best individual DR are denoted in bold. The results show that for certain problem instances the ensembles performed equally well or even better than the best individual DR in the ensemble. For the other instances it is evident that the ensemble did not obtain better results than the best individual DR. However, the ensemble always achieves a performance which is closer to the performance of the better individual DRs. Out of the individual DRs none of those contained in the ensemble perform well across all the problem instances. Thus, by performing well across all the instances the ensemble can perform better than individual DRs.

By analysing the performance of individual DRs that form the ensemble it is evident that the ensemble consists out of DRs with different qualities. On the training set all DRs perform rather well, which is expected since they were evolved on it. In this case the ensemble performed similar as the individual DRs. However, on the test set the performance of certain DRs is quite bad when compared to the performance of the ensemble. Thus the strength of the ensemble becomes evident when it is applied on new problem instances.

Table 13 represents the performance of the ensemble, and the individual DRs out of which it consists of, on several selected problem instances which were also used for analysing the vote combination method. The behaviour of the ensemble is similar as it was with the sum combination method. The ensemble again achieves a performance equal to the best DRs, while on other instances it usually perform similar to the better individuals DRs in the ensemble. The constructed ensemble performs better than any of the individual DRs on either of the two problem sets.

Table 13: Performance analysis of the best ensemble generated by SEC and combined with the vote combination method

Problem instance index	Individual DR							Ensemble
	5	9	19	29	32	39	46	
3	0.438	0.283	0.505	0.283	0.329	0.268	0.260	0.283
5	0.199	0.237	0.237	0.191	0.191	0.237	0.191	0.191
8	0.581	0.602	0.644	0.605	0.856	0.582	0.759	0.601
17	0.833	0.909	0.845	0.723	1.047	0.817	0.903	0.730
27	0.100	0.100	0.100	0.015	0.046	0.100	0.100	0.015
29	0.223	0.182	0.182	0.182	0.182	0.182	0.223	0.182
33	0.000	0.003	0.003	0.003	0.003	0.003	0.000	0.003
38	0.334	0.364	0.370	0.397	0.347	0.432	0.529	0.370
53	0.774	0.699	0.699	0.732	0.765	0.850	0.681	0.699
57	0.369	0.412	0.433	0.408	0.364	0.449	0.440	0.409
Total fitness on all instances	15.49	15.39	16.09	16.33	16.44	15.68	16.02	15.07
Fitness on the training set	14.48	14.61	14.57	14.71	14.63	14.81	14.64	14.46

7 Conclusion

In this paper the performance of the SEC method, which creates ensembles of automatically generated DRs, has been analysed with regards to different ensemble creation methods, ensemble parameters, approaches used to generate DRs and ensembles. The main objective of this paper was to obtain further insights of SEC and to validate whether it performs well in different situations. Five methods for constructing ensembles were tested in the paper, and the obtained results show that the more complicated ensemble construction methods do not necessarily lead to significantly better results. Furthermore, it was demonstrated that the construction methods can perform well even if a small number of ensembles are created during construction. This is an important conclusion since it allows that the computation complexity of the construction methods can be significantly reduced without any significant deterioration in the results. Further experiments demonstrated that the method works well on DRs which are generated by different GP variants. Finally, the results demonstrate that SEC works better with the vote combination method, since in that case it more consistently achieved good results. All things considered, it can be concluded that SEC can generate ensembles which improve the performance of DRs generated by different methods on several scheduling criteria. However this does come come with a drawback that they require a considerably larger amount of time to be constructed than a single DR.

Although this paper closely examined SEC, there are still areas which could be examined further. For example, it would be interesting to adapt this approach in static scheduling environments, so that the approach can quickly construct ensembles which will perform well for the given problem instances. Additionally, even though this paper has analysed several ensemble construction methods, it is possible to define new ensemble construction methods which could create even better ensembles in less time. Finally, it would also be in-

teresting to analyse how this approach would perform on different scheduling problems, or even other types of combinatorial optimisation problems.

References

1. Allahverdi, A., Gupta, J.N., Aldowaisan, T.: A review of scheduling research involving setup considerations. Omega **27**(2), 219–239 (1999). DOI 10.1016/S0305-0483(98)00042-5
2. Allahverdi, A., Ng, C., Cheng, T., Kovalyov, M.Y.: A survey of scheduling problems with setup times or costs. European Journal of Operational Research **187**(3), 985–1032 (2008). DOI 10.1016/j.ejor.2006.06.060
3. Branke, J., Groves, M.J., Hildebrandt, T.: Evolving control rules for a dual-constrained job scheduling scenario. In: Proceedings of the 2016 Winter Simulation Conference, WSC '16, pp. 2568–2579. IEEE Press, Piscataway, NJ, USA (2016)
4. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated Design of Production Scheduling Heuristics: A Review. IEEE Transactions on Evolutionary Computation **20**(1), 110–124 (2016). DOI 10.1109/TEVC.2015.2429314
5. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A Classification of Hyper-heuristic Approaches. In: Handbook of Metaheuristics, pp. 449–468 (2010). DOI 10.1007/978-1-4419-1665-5_15
6. Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: Exploring Hyper-heuristic Methodologies with Genetic Programming. Computational Intelligence **1**, 177–201 (2009). DOI doi:10.1007/978-3-642-01799-5_6
7. Cheng, V., Crawford, L., Menon, P.: Air traffic control using genetic search techniques. In: Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328), vol. 1, pp. 249–254. IEEE (1999). DOI 10.1109/CCA.1999.806209. URL http://ieeexplore.ieee.org/document/806209/
8. Dimopoulos, C., Zalzala, A.: Investigating the use of genetic programming for a classic one-machine scheduling problem. Research report, ARRAY(0x7f0faa5322f0) (1998). URL http://eprints.whiterose.ac.uk/82572/
9. Dimopoulos, C., Zalzala, A.: A genetic programming heuristic for the one-machine total tardiness problem. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), pp. 2207–2214. IEEE (1999). DOI 10.1109/CEC.1999.785549
10. Đurasević, M., Jakobović, D., Knežević, K.: Adaptive scheduling on unrelated machines with genetic programming. Applied Soft Computing **48**, 419 – 430 (2016)
11. Durasević, M., Jakobović, D.: Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. Genetic Programming and Evolvable Machines (2017). DOI 10.1007/s10710-017-9310-3. URL https://doi.org/10.1007/s10710-017-9310-3
12. Ferreira, C.: Gene expression programming: a new adaptive algorithm for solving problems. Complex Systems **13**(2), 87–129 (2001). URL http://arxiv.org/abs/cs/0102027
13. Greene, W.A.: Dynamic load-balancing via a genetic algorithm. In: Tools with Artificial Intelligence, Proceedings of the 13th International Conference on, pp. 121–128. IEEE (2001)
14. Hansen, J.V.: Genetic search methods in air traffic control. Computers & Operations Research **31**(3), 445–459 (2004). DOI 10.1016/S0305-0548(02)00228-9
15. Hart, E., Sim, K.: A Hyper-Heuristic Ensemble Method for Static Job-Shop Scheduling. Evolutionary Computation **24**(4), 609–635 (2016). DOI 10.1162/EVCO_a_00183
16. Hildebrandt, T., Heger, J., Scholz-Reiter, B.: Towards improved dispatching rules for complex shop floor scenarios. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10, p. 257. ACM Press, New York, New York, USA (2010). DOI 10.1145/1830483.1830530
17. Hunt, R., Johnston, M., Zhang, M.: Evolving "less-myopic" scheduling rules for dynamic job shop scheduling with genetic programming. In: Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14, pp. 927–934. ACM Press, New York, New York, USA (2014). DOI 10.1145/2576768.2598224

18. Hunt, R., Johnston, M., Zhang, M.: Evolving machine-specific dispatching rules for a two-machine job shop using genetic programming. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 618–625. IEEE (2014). DOI 10.1109/CEC.2014.6900655
19. Ingimundardottir, H., Runarsson, T.P.: Evolutionary Learning of Linear Composite Dispatching Rules for Scheduling. In: Computational Intelligence: International Joint Conference, pp. 49–62. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-26393-9_4
20. Jakobović, D., Budin, L.: Dynamic scheduling with genetic programming. In: P. Collet, M. Tomassini, M. Ebner, S. Gustafson, A. Ekárt (eds.) Genetic Programming: 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006. Proceedings, pp. 73–84. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
21. Jakobović, D., Marasović, K.: Evolving priority scheduling heuristics with genetic programming. Applied Soft Computing **12**(9), 2781–2789 (2012). DOI 10.1016/j.asoc.2012.03.065
22. Karunakaran, D., Chen, G., Zhang, M.: Parallel Multi-objective Job Shop Scheduling Using Genetic Programming. In: Artificial Life and Computational Intelligence: Second Australasian Conference, ACALCI 2016, Canberra, ACT, Australia, February 2-5, 2016, Proceedings, pp. 234–245. Springer International Publishing (2016). DOI 10.1007/978-3-319-28270-1_20
23. Keijzer, M.: Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. Genetic Programming Proceedings of EuroGP2003 **2610**, 70–82 (2003). DOI 10.1007/3-540-36599-0_7
24. Keijzer, M., Babovic, V.: Dimensionally Aware Genetic Programming. Proceedings of the Genetic and Evolutionary Computation Conference **2**, 1069–1076 (1999)
25. Koza, J.R.: Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines **11**(3-4), 251–284 (2010). DOI 10.1007/s10710-010-9112-3. URL http://link.springer.com/10.1007/s10710-010-9112-3
26. Lee, Y.H., Bhaskaran, K., Pinedo, M.: A heuristic to minimize the total weighted tardiness with sequence-dependent setups. IIE Transactions **29**(1), 45–52 (1997). DOI 10.1080/07408179708966311
27. Li, D., Zhan, R., Zheng, D., Li, M., Kaku, I.: A Hybrid Evolutionary Hyper-Heuristic Approach for Intercell Scheduling Considering Transportation Capacity. IEEE Transactions on Automation Science and Engineering **13**(2), 1072–1089 (2016). DOI 10.1109/TASE.2015.2470080. URL http://ieeexplore.ieee.org/document/7270346/
28. Mei, Y., Nguyen, S., Zhang, M.: Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling. In: Y. Shi, K.C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, Y. Jin (eds.) Simulated Evolution and Learning, pp. 435–447. Springer International Publishing, Cham (2017)
29. Mei, Y., Zhang, M., Nyugen, S.: Feature Selection in Evolving Job Shop Dispatching Rules with Genetic Programming. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16, pp. 365–372. ACM Press, New York, New York, USA (2016). DOI 10.1145/2908812.2908822
30. Miyashita, K.: Job-shop scheduling with genetic programming. In: Proceedings of the 2Nd Annual Conference on Genetic and Evolutionary Computation, GECCO'00, pp. 505–512. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2000)
31. Nguyen, S., Zhang, M., Johnston, M.: A sequential genetic programming method to learn forward construction heuristics for order acceptance and scheduling. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1824–1831. IEEE (2014). DOI 10.1109/CEC.2014.6900347
32. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. In: 2012 IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2012). DOI 10.1109/CEC.2012.6252968
33. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. IEEE Transactions on Evolutionary Computation **17**(5), 621–639 (2013). DOI 10.1109/TEVC.2012.2227326

34. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Dynamic multi-objective job shop scheduling: A genetic programming approach. In: A.S. Uyar, E. Ozcan, N. Urquhart (eds.) Automated Scheduling and Planning: From Theory to Practice, pp. 251–282. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
35. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Learning iterative dispatching rules for job shop scheduling with genetic programming. International Journal of Advanced Manufacturing Technology **67**(1-4), 85–100 (2013). DOI 10.1007/s00170-013-4756-9
36. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Automatic Design of Scheduling cies for Dynamic Multi-objective Job Shop Scheduling via Cooperative Coevolution Genetic Programming. IEEE Transactions on Evolutionary Computation **18**(2), 193–208 (2014). DOI 10.1109/TEVC.2013.2248159
37. Nguyen, S., Zhang, M., Tan, K.C.: A Dispatching rule based Genetic Algorithm for Order Acceptance and Scheduling. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15, pp. 433–440. ACM Press, New York, New York, USA (2015). DOI 10.1145/2739480.2754821
38. Nguyen, S., Zhang, M., Tan, K.C.: Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 2781–2788. IEEE (2015). DOI 10.1109/CEC.2015.7257234
39. Nie, L., Gao, L., Li, P., Zhang, L.: Application of gene expression programming on dynamic job shop scheduling problem. In: Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 291–295. IEEE (2011). DOI 10.1109/CSCWD.2011.5960088
40. Nie, L., Shao, X., Gao, L., Li, W.: Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. The International Journal of Advanced Manufacturing Technology **50**(5-8), 729–747 (2010). DOI 10.1007/s00170-010-2518-5
41. Park, J., Mei, Y., Nguyen, S., Chen, G., Zhang, M.: An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. Applied Soft Computing **63**, 72 – 86 (2018). DOI https://doi.org/10.1016/j.asoc.2017.11.020. URL http://www.sciencedirect.com/science/article/pii/S156849461730683X
42. Park, J., Nguyen, S., Zhang, M., Johnston, M.: Genetic programming for order acceptance and scheduling. In: 2013 IEEE Congress on Evolutionary Computation, pp. 1005–1012. IEEE (2013). DOI 10.1109/CEC.2013.6557677
43. Park, J., Nguyen, S., Zhang, M., Johnston, M.: Evolving ensembles of dispatching rules using genetic programming for job shop scheduling. In: Genetic Programming: 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings, pp. 92–104. Springer International Publishing, Cham (2015)
44. Petrovic, S., Castro, E.: A genetic algorithm for radiotherapy pre-treatment scheduling. In: Applications of Evolutionary Computation: EvoApplications 2011: EvoCOMNET, EvoFIN, EvoHOT, EvoMUSART, EvoSTIM, and EvoTRANSLOG, Torino, Italy, April 27-29, 2011, Proceedings, Part II, pp. 454–463. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
45. Pickardt, C.W., Hildebrandt, T., Branke, J., Heger, J., Scholz-Reiter, B.: Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. International Journal of Production Economics **145**(1), 67–77 (2013). DOI 10.1016/j.ijpe.2012.10.016
46. Pinedo, M.L.: Scheduling: Theory, algorithms, and systems: Fourth edition, vol. 9781461423614. Springer US, Boston, MA (2012). DOI 10.1007/978-1-4614-2361-4. URL http://link.springer.com/10.1007/978-1-4614-2361-4
47. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via `http://lulu.com` and available at `http://www.gp-field-guide.org.uk` (2008)
48. Polikar, R.: Ensemble learning. Scholarpedia **4**(1), 2776 (2009). DOI 10.4249/scholarpedia.2776. Revision #91224
49. Tay, J.C., Ho, N.B.: Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Computers & Industrial Engineering **54**(3), 453–473 (2008). DOI 10.1016/j.cie.2007.08.008

50. Đurasević, M., Jakobović, D.: Comparison of solution representations for scheduling in the unrelated machines environment. 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) pp. 1336 – 1342 (2006). DOI 10.1109/MIPRO.2016.7522347
51. Đurasević, M., Jakobović, D.: Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. Genetic Programming and Evolvable Machines pp. 1–40 (2017). DOI 10.1007/s10710-017-9302-3
52. Wang, X., Nie, L., Bai, Y.: Discovering scheduling rules with a machine learning approach based on GEP and PSO for dynamic scheduling problems in shop floor. In: Computational Intelligence in Industrial Application, pp. 365–370. CRC Press (2015). DOI 10.1201/b18590-71. URL http://www.crcnetbase.com/doi/10.1201/b18590-71

Appendix

A Problem instance details

The scheduling problem which is solved in this paper can be classified as $Rm|r_j|\gamma$, where γ represents one of the four scheduling criteria that are considered. In the unrelated machines scheduling problem each job needs to be scheduled on a single machine. When the job is scheduled on a certain machine, the machine needs to execute this job until it is finished before it can start executing another job (thus preemption is not permitted). At each moment in the each machine can execute at most one job. However, all machines work in parallel, which means that each of them is executing the job assigned to it, independently from the other machines. The unrelated machines environment is a single stage environment which means that each job needs to be executed on only a single machine to be completed. Furthermore, the specificness of this environment is that each job has a different execution time on each of the machines, thus selecting the appropriate machine for each job constitutes an important part in this problem. Except for the release times, no additional constraints, like breakdowns or set-up times, are considered in these problems. However, the propose SEC method should work with any of those without any changes, if the DRs are adapted for these additional changes. The way in which the problem instances are generated, the system will have a high utilisation most of the time during its execution.

The processing times of jobs are generated from the interval

$$p_{ij} \in [0, 100],$$

by using one of the following three probabilistic distributions: uniform, normal (Gaussian), and quasi-bimodal. Which of the aforementioned three distributions will be used for generating the processing times is chosen randomly for each job (with all three distributions having the same probability of being chosen). The motivation behind the use of three distributions for generating processing times is to make the evolved priority functions more resilient, since in real conditions jobs could be received from different sources. All job weights are generated uniformly from the interval

$$w_{T_j} \in < 0, 1].$$

The release times of jobs are generated by a uniform distribution from the interval

$$r_j \in \left[0, \frac{\hat{p}}{2}\right],$$

where $\hat{p}$ is defined as

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

and p_{ij} denotes the processing time of job j on machine i, while m denotes the total number of machines. The due dates of jobs are also defined using a uniform distribution from the interval

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

where T represents the due date tightness parameter, while R represents the due date range parameter. The due date range parameter defines the dispersion of the due date values, while the due date tightness adjusts the amount of jobs that will be late. While generating the problem set, both of those parameters assumed values of 0.2, 0.4, 0.6, 0.8, and 1 in various combinations.

Because some problem instances have significantly different characteristics, and will therefore also have significantly different objective values. This leads to a problem in which smaller instances have little or no influence in the total fitness value, and thus the GP procedure would focus less on optimising these instances. To avoid this problem all the objective values were normalised in order for the problem instances with different characteristics to have similar objective values. Therefore, the normalised objective functions for the problem instance with the index i are defined as follows:

- for the weighted tardiness criterion $f_i = \frac{\sum_{j=1}^{n} w_j T_j}{n\bar{w}\bar{p}}$
- for the weighted number of tardy jobs criterion $f_i = \frac{\sum_{j=1}^{n} w_j U_j}{n\bar{w}}$
- for the flowtime criterion $f_i = \frac{\sum_{j=1}^{n} F_j}{n\bar{p}}$
- for the makespan criterion $f_i = \frac{\max\{C_j\}}{n\bar{p}}$,

where n denotes the number of jobs in the problem instance, $\bar{w}$ the average weight of the jobs and $\bar{p}$ the average job processing duration. The total objective function is then calculated as the sum of the objective functions of the individual problem instances.

B Results for the ensemble construction methods on different criteria

In this section the results of the ensemble construction methods will be presented on the remaining three scheduling criteria. Table 14 represents the results obtained the ensemble construction methods for the Nwt criterion when

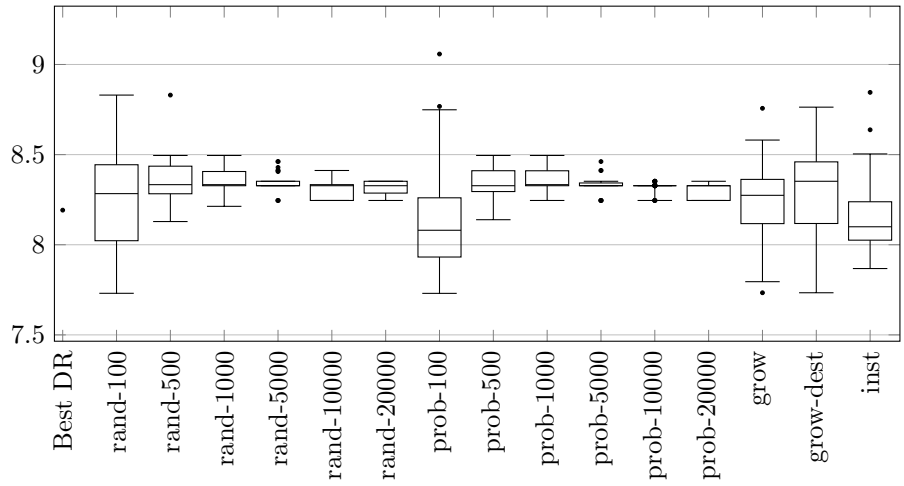
Table 14: Performance of the ensemble construction methods on the Nwt criterion with the sum combination method

Procedure	Ensemble size								
	3			5			7		
	Min	Med	Std	Min	Med	Std	Min	Med	Std
Best DR		8.192			8.192		8.192	8.192	
Rand 100	7.731	8.307	0.307	7.724	8.205	0.225	**7.546**	8.099	0.275
Rand 500	8.129	8.333	0.127	7.824	8.144	0.154	7.634	8.175	0.300
Rand 1000	8.213	8.333	0.074	7.750	8.257	0.211	7.649	8.175	0.249
Rand 5000	8.246	8.328	0.053	7.867	8.252	0.126	7.594	8.128	0.164
Rand 10000	8.246	8.308	0.048	7.756	8.251	0.142	7.877	8.166	0.110
Rand 20000	8.246	8.328	0.041	7.992	8.338	0.103	7.888	8.127	0.106
Prob 100	**7.730**	**8.080**	0.330	7.798	8.218	0.304	7.701	8.235	0.176
Prob 500	8.139	8.330	0.088	7.813	8.172	0.161	7.733	8.055	0.222
Prob 1000	8.246	8.333	0.066	7.720	8.116	0.216	7.803	8.140	0.208
Prob 5000	8.246	8.328	0.046	7.708	8.249	0.159	7.794	8.160	0.109
Prob 10000	8.246	8.328	**0.032**	8.042	8.230	0.126	7.796	8.135	0.122
Prob 20000	8.246	8.328	0.040	8.125	8.323	**0.093**	7.995	8.176	**0.083**
Grow	7.734	8.278	0.211	**7.571**	8.277	0.227	7.571	8.256	0.235
Grow-dest	7.734	8.352	0.230	**7.571**	8.256	0.235	7.645	8.230	0.240
Inst	7.868	8.100	0.203	7.777	**8.052**	0.152	7.606	**8.011**	0.162

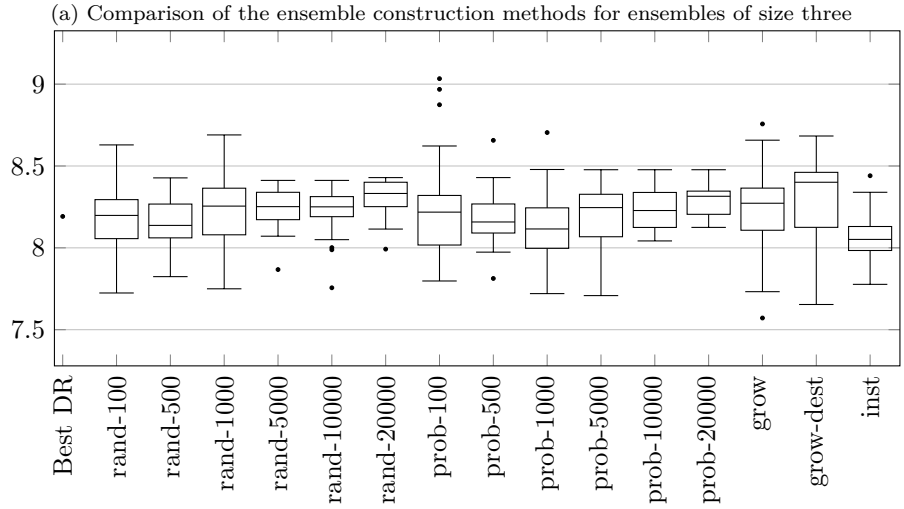
using the sum combination method. The box plot representation of these results is given in Figure 7. The results denote that the instance based method is the only one which consistently performed better on average than the best DR. The other methods performed well only when the ensemble size of 7 DRs was used, while in other cases the average results that were obtained were worse than the best DR. The box plots show that for the smallest ensemble size the performance of the ensembles is usually much worse than that of the best individual DRs. However, for the largest ensemble size it is evident that for certain experiments most of the ensembles perform better than the best DR.

Table 15 denotes the same results using the vote combination method. Figure 8 represents the results presented by using box-plots. In this case all the ensemble combination methods obtained average results which are better than the result obtained by the best DR. The results are quite similar, and between most of them there is no statistically significant difference. The box plot shows that most of the ensembles which were generated by SEC actually achieve a better performance than the best individual DR.

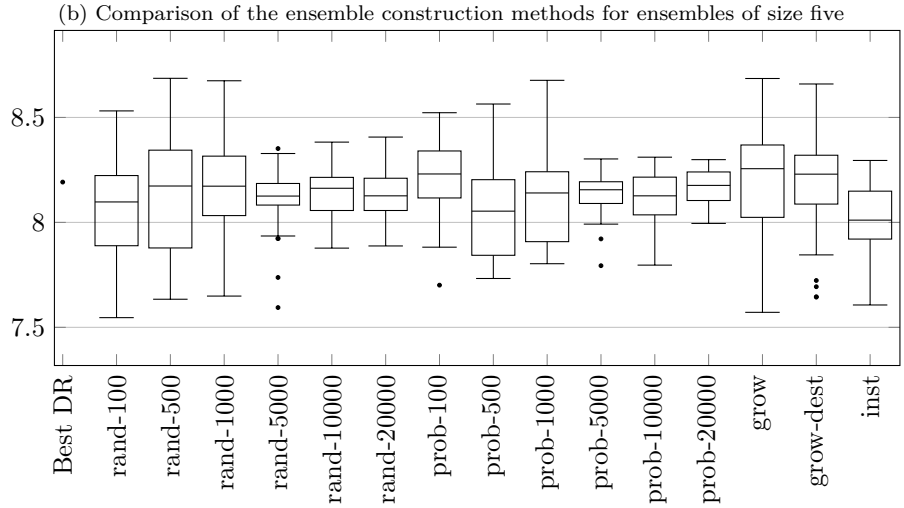
Table 16 represents the results obtained for the Ft criterion when using the sum combination method. The box-plot representations of the results are presented in Figure 9. The table shows some interesting results for this criterion. Namely, the results depend heavily on the size of the constructed ensemble. For the two smaller ensemble sizes the SEC method was unable to obtain better results on average than those obtained by the best DR. However, for the ensemble size 7 almost all tested ensemble construction methods obtained



(a) Comparison of the ensemble construction methods for ensembles of size three

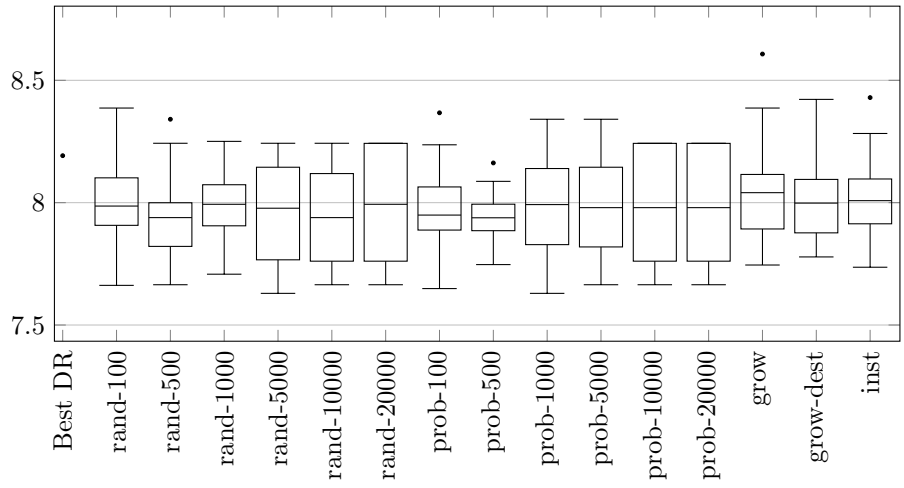


(b) Comparison of the ensemble construction methods for ensembles of size five

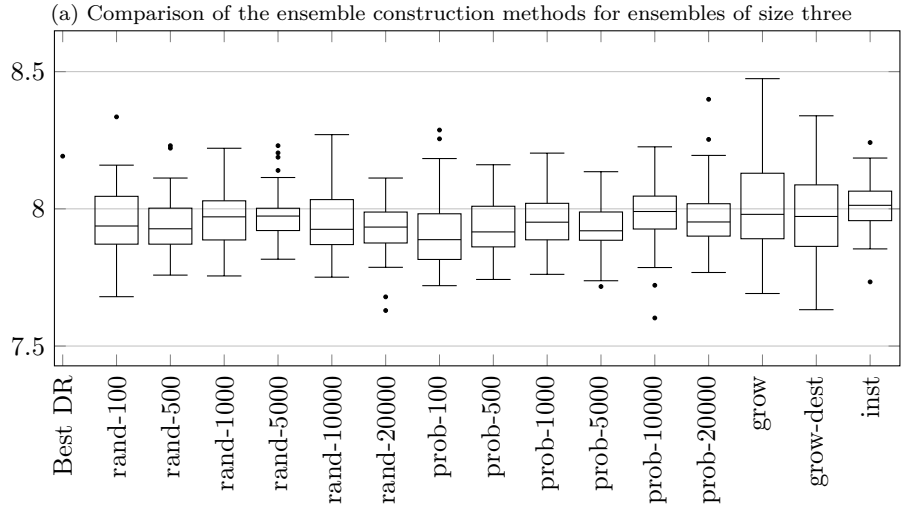


(c) Comparison of the ensemble construction methods for ensembles of size seven

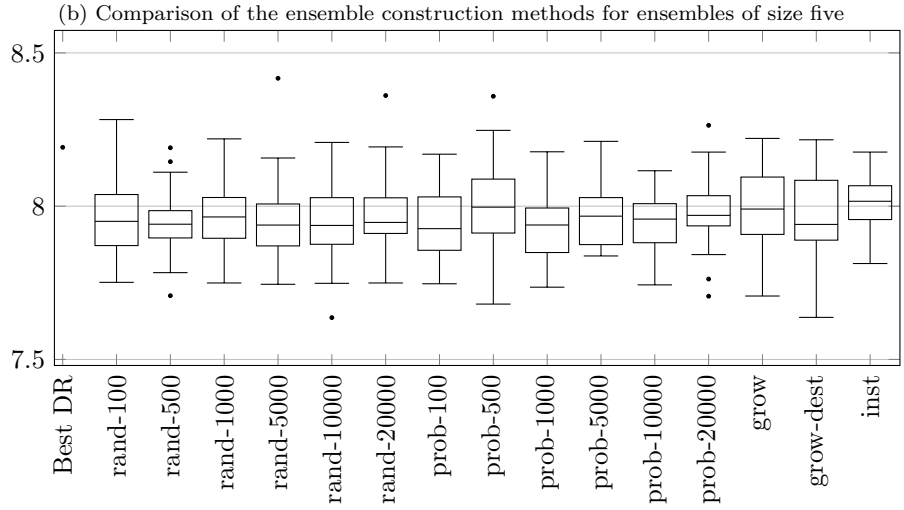
Fig. 7: Box-plot representation of results for the ensemble construction approaches and the sum combination method on the Nwt criterion



(a) Comparison of the ensemble construction methods for ensembles of size three



(b) Comparison of the ensemble construction methods for ensembles of size five



(c) Comparison of the ensemble construction methods for ensembles of size seven

Fig. 8: Box-plot representation of results for the ensemble construction approaches and the vote combination method on the Nwt criterion

Table 15: Performance of the ensemble construction methods on the Nwt criterion with the vote combination method

Procedure	Ensemble size								
	3			5			7		
	min	med	std	min	med	std	min	med	std
Best DR		8.192			8.192		8.192	8.192	
Rand 100	7.662	7.988	0.178	7.680	7.939	0.134	7.751	7.953	0.112
Rand 500	7.665	7.941	0.162	7.759	7.931	0.116	7.708	7.943	0.100
Rand 1000	7.708	7.995	0.122	7.755	7.975	0.114	7.749	7.968	0.109
Rand 5000	7.629	7.978	0.200	7.816	7.975	0.102	7.745	7.945	0.128
Rand 10000	7.665	**7.939**	0.206	7.751	7.942	0.122	**7.636**	7.942	0.123
Rand 20000	7.665	8.069	0.230	7.630	7.937	0.108	7.749	7.948	0.113
Prob 100	**7.649**	7.959	0.156	7.720	**7.889**	0.149	7.747	**7.927**	0.116
Prob 500	7.747	7.943	**0.094**	7.743	7.916	0.106	7.681	7.998	0.146
Prob 1000	7.629	7.994	0.185	7.761	7.952	0.113	7.736	7.941	0.106
Prob 5000	7.665	7.980	0.205	7.717	7.920	0.101	7.838	7.969	0.096
Prob 10000	7.665	7.986	0.219	**7.602**	8.007	0.128	7.743	7.959	0.102
Prob 20000	7.665	7.980	0.231	7.768	7.958	0.134	7.706	7.972	0.106
Grow	7.745	8.042	0.169	7.691	7.987	0.180	7.707	7.993	0.131
Grow-dest	7.778	8.001	0.142	7.632	7.976	0.152	7.637	7.945	0.130
Inst	7.736	8.010	0.145	7.734	8.016	**0.092**	7.813	8.018	**0.085**

better results than the best DR. The best construction method seems to be the random selection method in this case.

Table 17 shows the results obtained when optimising the Ft criterion and using the vote combination method. The box-plots of the results are presented in Figure 10. Similarly as with the Nwt criterion, all of the results obtained by the ensemble combination methods are better than those obtained by the best DR. The box-plots denote that the different ensemble combination methods achieve results with no significant difference between them. The constructed ensembles were once again batter in most cases than the best individual DR.

Table 18 shows the results obtained with the sum combination method when optimising the C_{max} criterion. Figure 11 shows the box-plot representation of the results. Most of the results obtained by the SEC method are not better than those obtained by individual DRs. Only in very few cases the ensembles were actually better on average than the best DR.

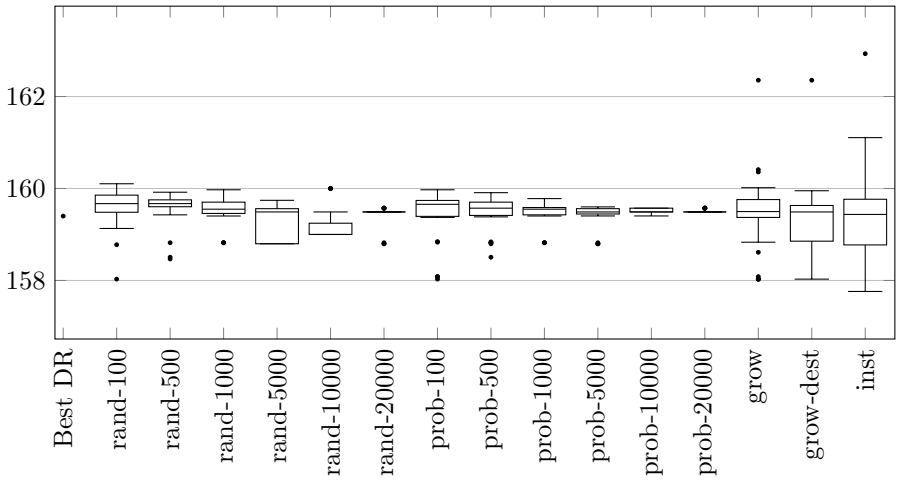
Table 19 shows the results obtained when optimising the C_{max} criterion and when using the vote combination method. Figure 12 represent the results in box-plots. All ensemble combination methods obtain better results on average than the individual DRs. The only exception is the instance based method which constantly achieved quite bad results in comparison with the individual DRs. The differences between the different ensemble construction methods are usually small and not significantly different. The box plots show that for ensemble sizes 5 and 7 the SEC method has once again proven to achieve better results when it is used with the vote combination method. In addition,

Table 16: Performance of the ensemble construction methods on the Ft criterion with the sum combination method

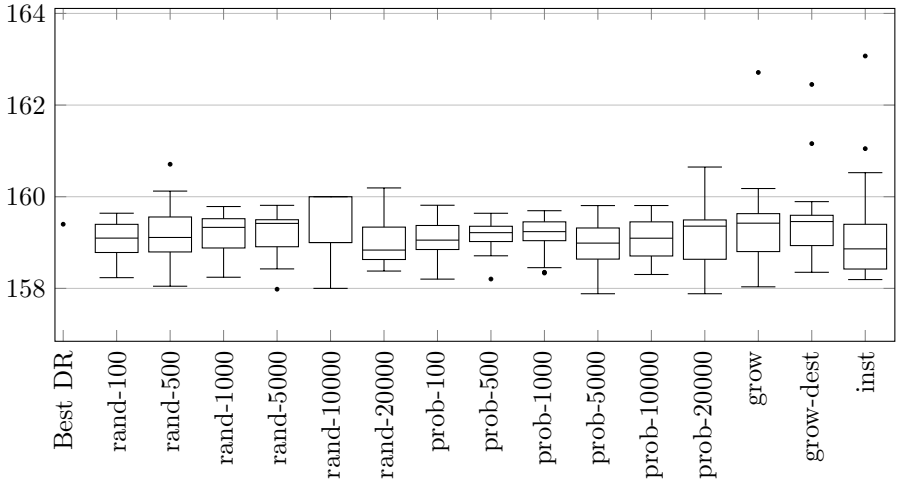
Procedure	Ensemble size								
	3			5			7		
	min	med	std	min	med	std	min	med	std
Best DR		159.4			159.4			159.4	
Rand 100	158.0	159.7	0.423	158.2	159.1	0.387	157.9	159.0	0.425
Rand 500	158.5	159.7	0.357	158.0	159.1	0.577	157.9	159.1	0.456
Rand 1000	158.8	159.6	0.278	158.2	159.3	0.420	158.2	159.1	0.477
Rand 5000	158.8	**159.5**	0.344	158.0	159.4	0.436	158.2	158.8	0.486
Rand 10000	158.8	**159.5**	0.304	158.2	159.3	0.471	158.0	158.9	0.420
Rand 20000	158.8	**159.5**	0.216	158.4	**158.8**	0.453	157.9	**158.7**	0.419
Prob 100	158.0	159.7	0.537	158.2	159.1	0.445	158.0	159.1	0.449
Prob 500	158.5	159.6	0.362	158.2	159.2	**0.319**	158.1	158.8	**0.341**
Prob 1000	158.8	159.6	0.251	158.3	159.3	0.367	158.0	158.8	0.499
Prob 5000	158.8	**159.5**	0.251	**157.9**	159.0	0.493	158.2	158.8	0.417
Prob 10000	159.4	**159.5**	0.049	158.3	159.2	0.461	158.1	158.8	0.387
Prob 20000	159.5	**159.5**	**0.033**	**157.9**	159.4	0.591	158.2	159.0	0.624
Grow	158.0	**159.5**	0.709	158.0	159.4	0.759	**157.8**	159.3	0.785
Grow-dest	158.0	**159.5**	0.700	158.4	159.5	0.656	158.1	158.9	0.722
Inst	**157.8**	**159.5**	0.963	158.2	158.9	0.868	158.4	159.2	0.559

Table 17: Performance of the ensemble construction methods on the Ft criterion with the vote combination method

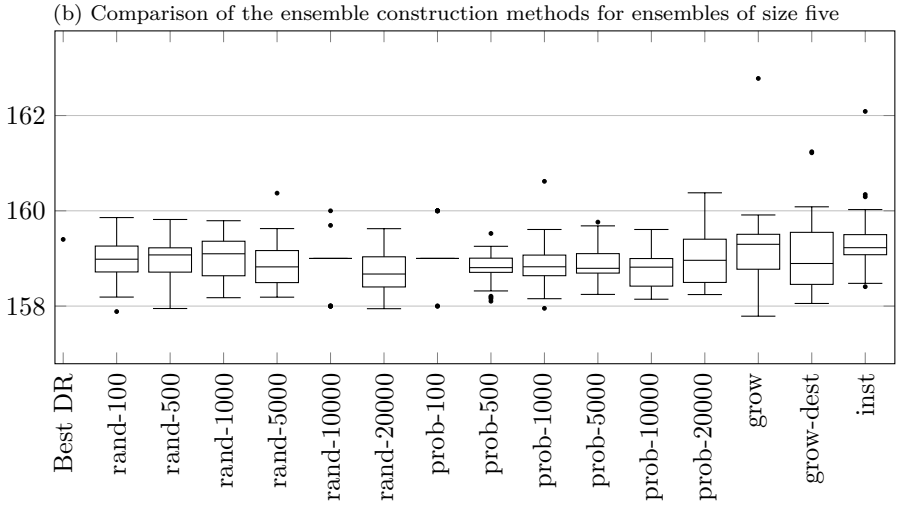
Procedure	Ensemble size								
	3			5			7		
	min	med	std	min	med	std	min	med	std
Best DR		159.4			159.4			159.4	
Rand 100	**157.5**	158.8	0.663	158.1	158.9	0.449	157.9	158.7	0.446
Rand 500	157.8	158.8	0.584	157.8	158.7	0.437	157.7	158.7	0.503
Rand 1000	158.0	158.8	0.459	**157.5**	158.7	0.595	157.3	158.6	0.508
Rand 5000	157.9	159.0	0.633	157.6	159.1	0.727	156.8	158.8	0.861
Rand 10000	158.3	**158.6**	0.310	157.6	158.8	0.568	157.5	158.6	0.507
Rand 20000	158.3	158.7	0.279	158.1	158.7	0.472	158.0	158.6	0.394
Prob 100	157.6	158.7	0.540	157.8	158.8	0.471	157.8	158.7	0.430
Prob 500	158.0	158.7	0.511	157.9	158.7	0.528	157.7	158.5	0.448
Prob 1000	158.0	158.7	0.686	158.0	158.7	0.515	157.8	158.6	0.444
Prob 5000	158.4	158.7	0.268	157.9	158.9	0.482	158.1	158.6	**0.375**
Prob 10000	158.6	158.7	0.052	158.0	158.7	**0.280**	**157.2**	158.6	0.578
Prob 20000	158.6	158.7	**0.045**	158.2	159.1	0.352	157.7	158.6	0.500
Grow	**157.5**	158.9	0.764	**157.7**	158.8	0.584	157.4	158.6	0.591
Grow-dest	157.6	158.9	0.768	**157.5**	158.8	0.517	157.6	158.5	0.555
Inst	157.7	159.1	0.793	157.6	**158.6**	0.430	157.8	**158.4**	0.437



(a) Comparison of the ensemble construction methods for ensembles of size three

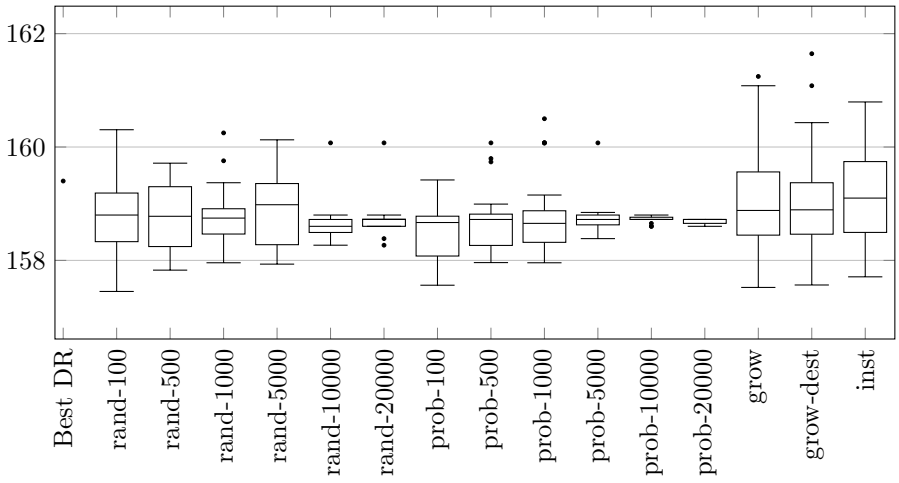


(b) Comparison of the ensemble construction methods for ensembles of size five

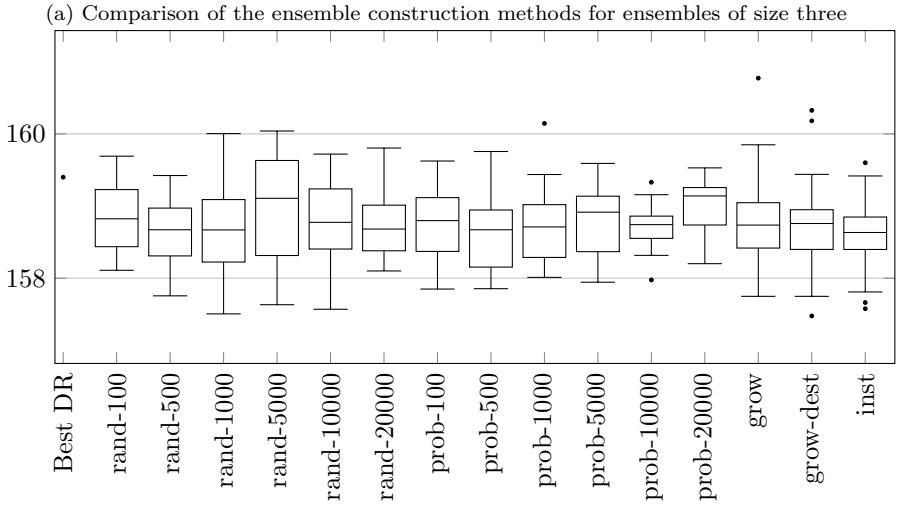


(c) Comparison of the ensemble construction methods for ensembles of size seven

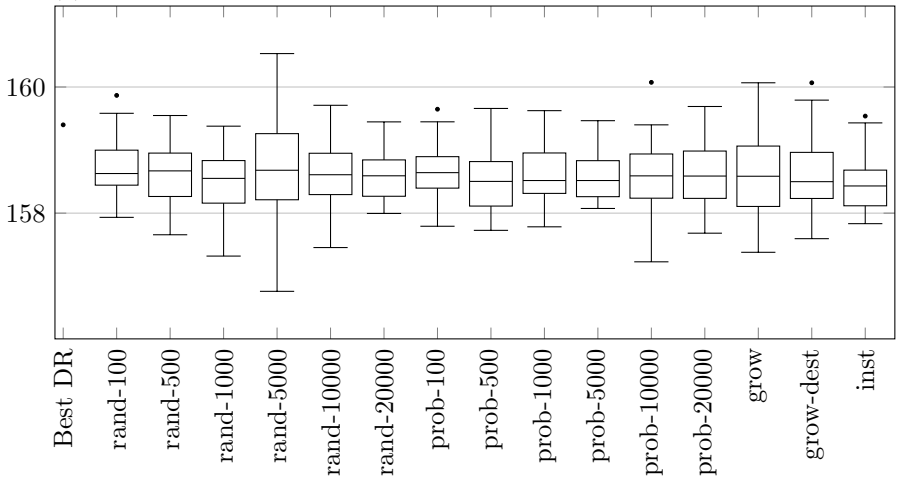
Fig. 9: Box-plot representation of results for the ensemble construction approaches and the sum combination method on the Ft criterion



(a) Comparison of the ensemble construction methods for ensembles of size three

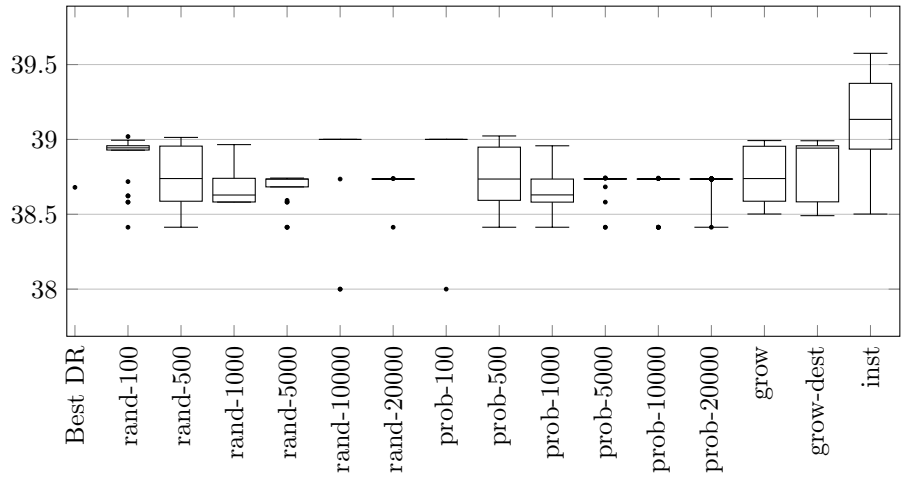


(b) Comparison of the ensemble construction methods for ensembles of size five

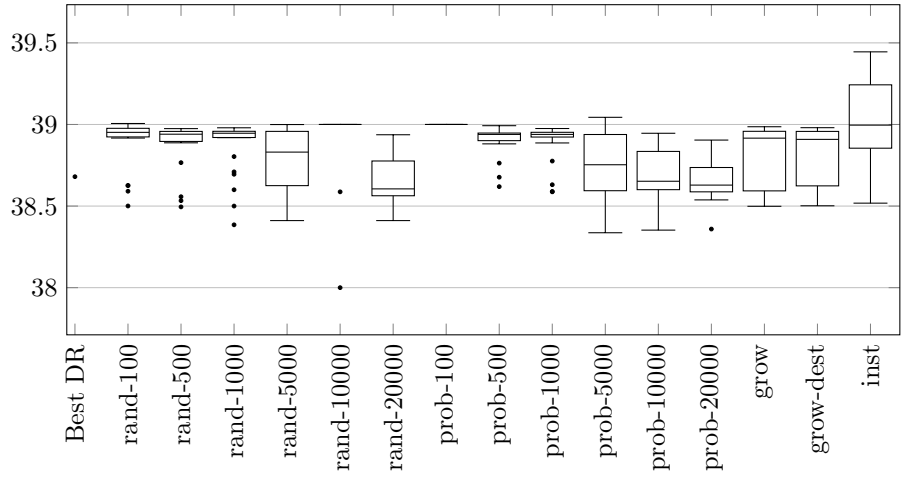


(c) Comparison of the ensemble construction methods for ensembles of size seven

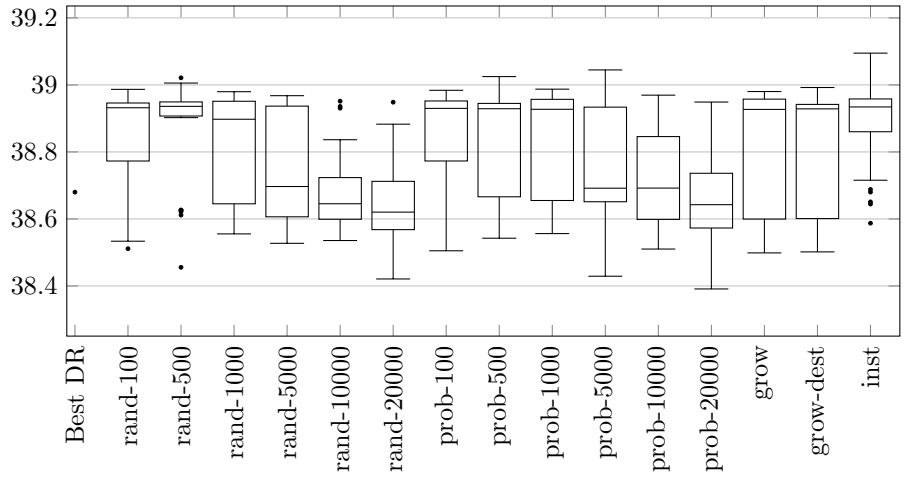
Fig. 10: Box-plot representation of results for the ensemble construction approaches and the vote combination method on the Ft criterion



(a) Comparison of the ensemble construction methods for ensembles of size three



(b) Comparison of the ensemble construction methods for ensembles of size five



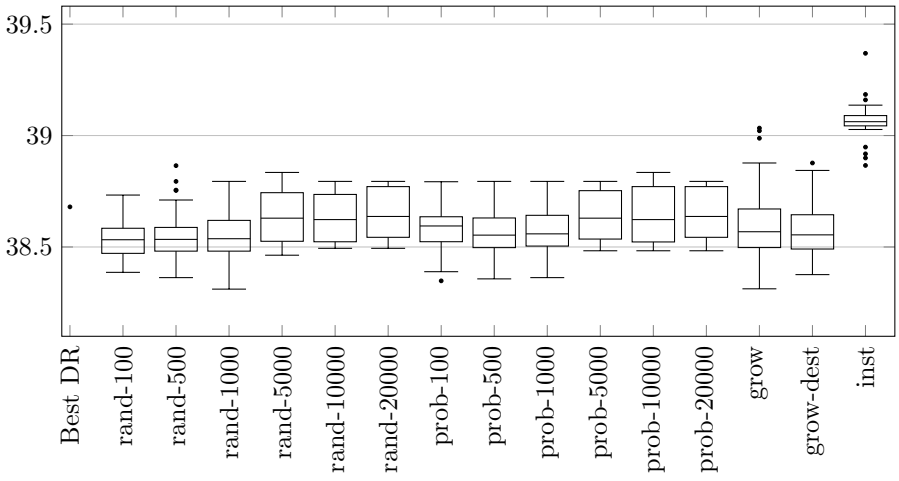
(c) Comparison of the ensemble construction methods for ensembles of size seven

Fig. 11: Box-plot representation of results for the ensemble construction approaches and the sum combination method on the C_{max} criterion

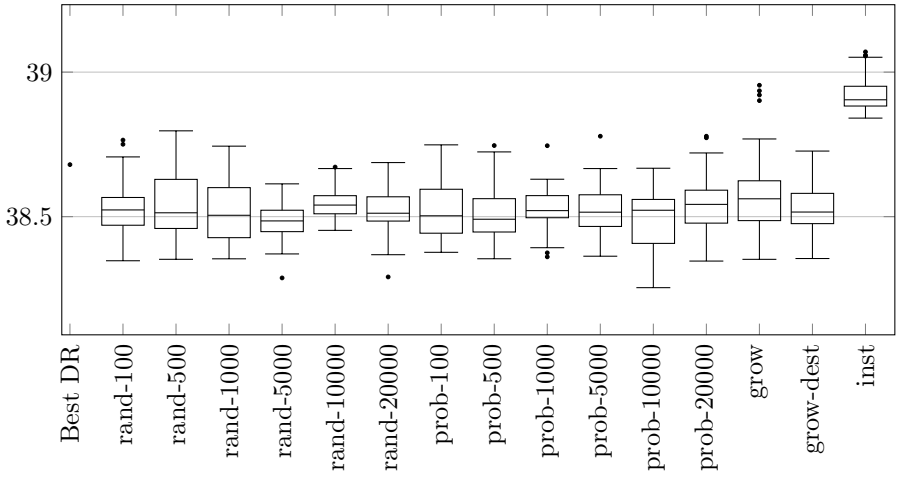
Table 18: Performance of the ensemble construction methods on the C_{max} criterion with the sum combination method

Procedure	Ensemble size								
	3			5			7		
	min	med	std	min	med	std	min	med	std
Best DR		38.68			38.68			38.68	
Rand 100	**38.41**	38.94	0.156	38.50	38.95	0.133	38.51	38.93	0.153
Rand 500	**38.41**	38.74	0.178	38.50	38.94	0.130	38.46	38.94	0.149
Rand 1000	38.58	**38.63**	0.149	38.39	38.95	0.151	38.56	38.90	0.148
Rand 5000	**38.41**	38.74	0.103	38.41	38.84	0.180	38.53	38.72	0.166
Rand 10000	**38.41**	38.74	0.082	**38.34**	38.71	0.166	38.54	38.65	0.114
Rand 20000	**38.41**	38.74	**0.059**	38.41	**38.63**	0.158	38.42	**38.63**	0.116
Prob 100	**38.41**	38.84	0.182	38.50	38.93	0.166	38.51	38.93	0.141
Prob 500	**38.41**	38.74	0.187	38.62	38.94	**0.083**	38.54	38.93	0.152
Prob 1000	**38.41**	**38.63**	0.111	38.59	38.94	0.108	38.56	38.93	0.149
Prob 5000	**38.41**	38.74	0.085	**38.34**	38.76	0.189	38.43	38.78	0.166
Prob 10000	**38.41**	38.74	0.122	38.35	38.68	0.154	38.51	38.70	0.141
Prob 20000	**38.41**	38.74	**0.059**	38.36	**38.63**	0.119	**38.39**	38.64	0.130
Grow	38.50	38.83	0.174	38.50	38.92	0.183	38.50	38.93	0.178
Grow-dest	38.49	38.94	0.181	38.50	38.92	0.174	38.50	38.93	0.182
Inst	38.50	39.14	0.259	38.52	39.00	0.205	38.59	38.93	**0.110**

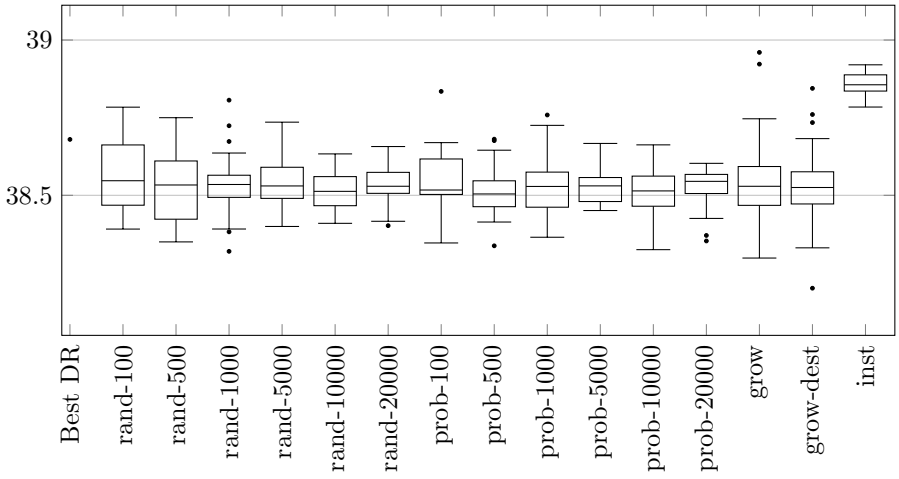
the box-plots also show that most of the constructed ensembles achieve a better performance than the best individual DR.



(a) Comparison of the ensemble construction methods for ensembles of size three



(b) Comparison of the ensemble construction methods for ensembles of size five



(c) Comparison of the ensemble construction methods for ensembles of size seven

Fig. 12: Box-plot representation of results for the ensemble construction approaches and the vote combination method on the C_{max} criterion

Table 19: Performance of the ensemble construction methods on the C_{max} criterion with the vote combination method

Procedure	Ensemble size								
	3			5			7		
	min	med	std	min	med	std	min	med	std
Best DR		38.68			38.68			38.68	
Rand 100	38.39	**38.53**	0.090	38.35	38.52	0.100	38.39	38.55	0.117
Rand 500	38.36	38.54	0.121	38.35	38.52	0.113	38.35	38.53	0.115
Rand 1000	**38.31**	38.54	0.122	38.35	38.51	0.114	38.32	38.54	0.098
Rand 5000	38.46	38.63	0.116	38.29	**38.49**	0.066	38.40	38.53	0.074
Rand 10000	38.49	38.63	0.114	38.45	38.54	**0.056**	38.41	38.52	0.059
Rand 20000	38.49	38.64	0.110	38.29	38.53	0.080	38.40	38.53	0.061
Prob 100	38.35	38.60	0.103	38.38	38.50	0.097	38.35	38.52	0.099
Prob 500	38.36	38.56	0.104	38.35	38.50	0.100	38.34	**38.50**	0.077
Prob 1000	38.36	38.56	0.096	38.36	38.52	0.080	38.36	38.53	0.093
Prob 5000	38.48	38.63	0.102	38.36	38.52	0.090	38.45	38.53	0.055
Prob 10000	38.48	38.63	0.122	**38.25**	38.52	0.107	38.32	38.52	0.073
Prob 20000	38.48	38.64	0.118	38.35	38.55	0.106	38.35	38.55	0.060
Grow	**38.31**	38.59	0.169	38.35	38.56	0.138	38.30	38.53	0.128
Grow-dest	38.38	38.56	0.132	38.36	38.53	0.085	**38.20**	38.53	0.105
Inst	38.87	39.06	**0.072**	38.84	38.90	0.064	38.78	38.86	**0.033**