Genet Program Evolvable Mach (2018) 19:53–92
https://doi.org/10.1007/s10710-017-9302-3

Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment

Marko Đurasević[1] · Domagoj Jakobović[1]

Received: 3 October 2016 / Revised: 12 March 2017 / Published online: 8 April 2017

Abstract Dispatching rules are often the method of choice for solving various scheduling problems, especially since they are applicable in dynamic scheduling environments. Unfortunately, dispatching rules are hard to design and are also unable to deliver results which are of equal quality as results achieved by different metaheuristic methods. As a consequence, genetic programming is commonly used in order to automatically design dispatching rules. Furthermore, a great amount of research with different genetic programming methods is done to increase the performance of the generated dispatching rules. In order to additionally improve the effectiveness of the evolved dispatching rules, in this paper the use of several different ensemble learning algorithms is proposed to create ensembles of dispatching rules for the dynamic scheduling problem in the unrelated machines environment. Four different ensemble learning approaches will be considered, which will be used in order to create ensembles of dispatching rules: simple ensemble combination (proposed in this paper), BagGP, BoostGP and cooperative coevolution. Additionally, the effectiveness of these algorithms is analysed based on some ensemble learning parameters. Finally, an additional search method, which finds the optimal combinations of dispatching rules to form the ensembles, is proposed and applied. The obtained results show that by using the aforementioned ensemble learning approaches it is possible to significantly increase the performance of the generated dispatching rules.

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

✉ Marko Đurasević
marko.durasevic@fer.hr

Domagoj Jakobović
domagoj.jakobovic@fer.hr

[1] Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia

1 Introduction

Scheduling can be defined as a decision making process concerned with the allocation of tasks to scarce resources with the intention of optimising one or more user defined scheduling objectives [40]. Although different approaches have been defined for solving various scheduling problems, dispatching rules represent the methods of choice when dealing with dynamic scheduling problems. Dispatching rules (DRs) usually represent a simple function which determines the priorities of jobs that need to be scheduled, and based on those priorities decides which job should be scheduled. They are very popular methods for solving scheduling problems since they can be designed to optimise various scheduling criteria, and can be used for different scheduling environments and conditions. Since designing good DRs usually represents a lengthy trial and error process, researchers have focused on defining procedures which could automatically design new dispatching rules.

In order to deal with the problem of manual design of DRs, many different machine learning methods were used in order to automatically create DRs [5]. One of the most commonly used procedures in the automatic development of DRs is genetic programming (GP) [24, 41]. By using GP it is possible to create DRs for a wide variety of different scheduling conditions and scheduling objectives. This feature becomes even more important when there is a need to design DRs for arbitrary user defined criteria, since DRs for such criteria might not even exist. Additionally, DRs generated by GP have in most cases been able to outperform manually designed DRs. Because GP is able to generate good DRs efficiently, in recent years a lot of research has been undertaken in order to apply GP for generating DRs for a wide variety of scheduling problems, as well as to improve the performance of the generated DRs.

This paper analyses if the performance of DRs generated by GP can be improved by using different ensemble learning approaches. The motivation for using ensemble learning approaches comes from the fact that, in the machine learning field, ensemble learning approaches have shown to improve the results achieved for various classification problems [42]. Four ensemble learning approaches will be considered: simple ensemble combination, BagGP, BoostGP and cooperative coevolution. For each of the considered approaches the influence of the ensemble size and the ensemble combination method on the results will be analysed. Additionally, for all the aforementioned approaches a further step, which tries to find a better subset of DRs that should form the ensemble, is introduced.

The remainder of this paper is organised as follows. Section 2 gives a short literature overview concerned with the automatic creation of DRs with GP. The unrelated machines environment is described in Sect. 3. Section 4 describes the GP procedure used in order to automatically create DRs, while Sect. 5 describes the ensemble learning approaches used in this paper. The results achieved by the ensemble learning approaches are outlined in Sect. 6. In Sect. 7 a short discussion about the achieved results is given. Finally, Sect. 8 gives a short conclusion and outlines possibilities for future work.

2 Literature overview

Since it is able to evolve quite complex expressions and functions, GP has been used in the field of hyper-heuristics quite often [7, 8]. Consequentially, GP is also used in order to evolve new DRs for different scheduling problems. One of the first uses of GP in scheduling was in order to generate a sequence in which existing DRs need to be applied in order to create the schedule [9]. Miyashita later evolved DRs for the job-shop environment by using GP with a terminal set that contained several job properties [25]. In his work, Miyashita considered the scheduling environment as a multi agent system where each machine represented an individual agent. Based on that he proposed three different models: the homogeneous model, the distinct agent model and the mixed agent model. The homogeneous model generated a single DR for all machines in the scheduling environment. On the other hand, the distinct agent model generated a distinct DR for each machine in the scheduling environment. Finally, the mixed agent model combines the two aforementioned models in a way that two DRs are evolved, first of which will be used by bottleneck machines, while the second will be used by all other machines. Although the mixed agent model achieved the best results among the three multi-agent models, it comes with an obvious disadvantage, which is that the knowledge about which machines are bottlenecks needs to be known before the system starts with its execution. In their work, Jakobović et al. propose a GP model which extends the mixed agent model from Miyashita. Their GP approach generates three expressions instead of one. Two of those expressions represent regular DRs, while the third expression represents a decision function which determines which of the two DRs will be used for a concrete machine. In that way there is no need to have prior knowledge about which machines represent bottleneck resources, but this can rather be determined during the system execution using the decision function. Apart from its application in the single machine and job-shop environments, GP was also used to create new DRs in the parallel machines environment with good results [22].

Unlike in the aforementioned works, where only a single optimisation criterion was considered, Tay and Ho have used GP to generate DRs which were designed to optimise three criteria at the same time [45]. Hildebrandt et al. have performed an extensive analysis on creating DRs for the job-shop environment [16]. Jakobović and Marasović have further investigated the creation of DRs for the single machine and job-shop environments [23]. In their work they analysed the influence of the GP parameters on the quality of the evolved DRs. Apart from that they also analysed scheduling in the single machine environment under various constraints like set-up times and precedence constraints, and have also shown that GP was able to achieve better results than some standard DRs. Gene expression programming [11], a method similar to GP, was also used in order to evolve DRs for both the single machine environment [35] and job-shop environment [34]. The problem of global perspective of DRs and how GP can be used in order to evolve DRs with a better global perspective was analysed in [18]. In a study by Hunt et al. it was shown that GP is able to evolve optimal DRs for the static two machine job-shop environment, which demonstrates that with the right parameters GP is able to evolve optimal DRs.

Different representations in GP were analysed by Nguyen et al. and it was shown that the representation used for evolving DRs influences the quality of the generated DRs [30]. A new GP approach which evolved iterative dispatching rules (IDRs) was proposed by Nguyen et al. [32]. Although the aforementioned approach was able to achieve better results than GP which evolves standard DRs, IDRs can only be used in the static environment in which information about the scheduling environment is known beforehand. Đurasevic et al. have compared several GP approaches for creating DRs in the unrelated scheduling environment, including GEP, IDRs and dimensionally aware GP [10]. Apart from generating DRs for the standard scheduling problems GP has also been applied in order to generate DRs for the order and acceptance (OAS) scheduling problem [26, 27, 37]. In the OAS problem, aside from scheduling jobs on machines, the system needs to decide which jobs will be accepted for scheduling. The generated DRs have also shown to be better than the standard DRs for the OAS problem, which shows that GP can generate DRs even for other forms of scheduling problems.

GP was also used in order to generate complete scheduling procedures (SPs), which consist of both DRs and due-date assignment rules (DDARs) [28, 33]. Those approaches used the cooperative coevolution procedure in order to generate two expressions (one of which represents a DR, while the other represents a DDAR) which together form a SP. The SPs evolved by GP have shown to be able to outperform some standard SPs from the literature. Nguyen et al. have used GP in order to generate DRs for optimising five scheduling criteria simultaneously and have shown that GP was able to evolve efficient DRs for the considered multi-objective criteria [29, 31]. A more in depth review of creating DRs by using GP can be found in [5].

Ensemble learning is often used in order to improve the performance of classifier systems [42]. Although ensemble learning approaches like bagging [6] or boosting [14] are commonly used in the machine learning community, ensemble learning approaches have not been as extensively used together with GP in order to improve its performance. Some notable applications of GP ensembles in the literature include classification with unbalanced data [3, 4], pattern classification [13] and intrusion detection [12]. GP ensemble learning approaches have been used for creating ensembles of DRs in only few occasions. In their work Park et al. [38] used the cooperative coevolution approach in order to create ensembles of DRs, and it was shown that such an approach achieves better results than standard GP. Unfortunately, in their work they only considered the static scheduling environment and did not additionally consider dynamic scheduling. Hart and Sim [15] propose a new hyper-heuristic called NELLI-GP which was used to solve static job-shop scheduling problems. This method creates an ensemble of DRs where each DR in the ensemble tries to adapt to a certain subset of problem instances.

3 Unrelated machines environment

The unrelated machines environment can be defined as a scheduling environment which consists of n jobs that need to be scheduled on one of the m available machines. Each machine can only execute one job at a time, and similarly, each job can only be executed by one machine. Preemption is not allowed, meaning that when a job starts executing on a given machine, it will execute until it is completed, after which a new job can be scheduled on the machine. Additionally, if release times are defined for jobs, then no job can start with execution before its respective release time. In this environment each job consists of several parameters:

- processing time p_{ij}—defines the time needed for job with the index j to be executed on machine with the index i
- release time r_j—defines the time in which the job with the index j becomes available
- due date d_j—defines the point in time until which the job with the index j should finish with its execution, otherwise a certain loss will be incurred
- weight w_j—defines the weight (importance) of the job with index j

After constructing the entire schedule, certain metrics are calculated for each job:

- C_j—finishing time of job j
- F_j—flowtime of job j:

$$F_j = C_j - r_j. \tag{1}$$

- T_j—tardiness of job j:

$$T_j = \max\{C_j - d_j, 0\}. \tag{2}$$

- U_j—flag if job is tardy or not:

$$U_j = \begin{cases} 1: & T_j > 0 \\ 0: & T_j = 0 \end{cases}. \tag{3}$$

Based on the previously defined job metrics, many different scheduling criteria can be defined [1, 2]. This study will focus on optimising the following four scheduling criteria:

- *Twt*—total weighted tardiness:

$$Twt = \sum_j w_j T_j, \tag{4}$$

- *Nwt*—weighted number of tardy jobs:

$$Nwt = \sum_j w_j U_j. \tag{5}$$

- *Ft*—total flowtime:

$$Ft = \sum_j F_j, \tag{6}$$

- C_{max}—maximum finish time of all jobs:

$$C_{max} = \max_j \{C_j\}. \tag{7}$$

Apart from the scheduling criteria which are optimised, it is also important to outline under which scheduling conditions the problem is solved. If all job parameters are available before the system starts with its execution, then this type of scheduling is called static scheduling. As a consequence, search-based methods (like genetic algorithms or ant colony optimisation) can be used in order to construct the schedule before the start of the system execution. On the other hand, if job parameters become available only as the jobs are released into the system, and no knowledge about their values is available beforehand, then this type of scheduling is called dynamic scheduling. Since there is a need to quickly adapt to the changing scheduling conditions, search-based methods most often cannot be used for this type of scheduling. Because of that reason, DRs are the most commonly used methods for creating schedules in dynamic environments, since they can quickly react to the changing environment. In this paper the dynamic scheduling environment is considered, in which job parameters become available only when the job is released, and the schedule is constructed together with the execution of the system. Therefore, since the schedule is constructed in parallel with the execution of the system it is important that the scheduling decision can be performed quickly in order to not incur any additional delay.

4 Creating DRs with GP

DRs which are constructed in this study can be divided into two parts: a meta-algorithm and a priority function (PF). The meta-algorithm defines a procedure which is used in order to create the entire schedule incrementally. Although the meta-algorithm defines a global scheduling procedure, it still needs to use a concrete PF which is used to calculate priority values for jobs and machines. These priority values are then used by the meta-algorithm in order to determine which job should be scheduled on which machine and in which order. Algorithm 1 represents the meta-algorithm which is used in this study. This procedure tries to find the best mapping between a job and a machine. If the machine on which the chosen job should be scheduled is available, then the job is immediately scheduled on that machine. On the other hand if the machine is currently busy and executing another job, then the job will not be scheduled, but the scheduling decision will be postponed to a later moment in time.

Algorithm 1 Meta-algorithm used for GP scheduling

```
1:  while unscheduled jobs are available do
2:      wait until a job becomes ready or a job finishes;
3:      for all available jobs and all machines do
4:          obtain the priority $\pi_{ij}$ of job $j$ on machine $i$;
5:      end for
6:      for all available jobs do
7:          determine the best machine (the one for which
8:          the best value of priority $\pi_{ij}$ is achieved);
9:      end for
10:     while jobs whose best machine is available exist
11:      do
12:         determine the best priority of all such jobs;
13:         schedule the job with the best priority;
14:     end while
15: end while
```

Unlike the meta-algorithm, which is manually defined, the PFs used by it are evolved by GP. However, in order for GP to be able to evolve quality DRs, relevant information about the scheduling environment and its current state, which will be available to GP in the evolution process, needs to be defined. This is done by specifying a set of terminal nodes which will be used by GP in the construction of DRs. Table 1 represents the set of selected terminal nodes which will be used in the evolution process. The *time* variable, which appears in the description of some terminal nodes, represents the current time of the system. Terminals *pt*, *dd* and *w* represent given properties of the jobs. The *pmin* and *pavg* terminals are included in order to give DRs the ability to determine whether the currently considered

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
TFMA	*Time until fastest machine available*—the amount of time until the machine with the minimal processing time for the current job will be available
TMR	*Time until machine is 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 due date related criteria (tardiness and number of tardy jobs)	
dd	Due date (d_j)
w	Weight (w_j)
SL	Positive slack: $max\{d_j - p_{ij} - time, 0\}$

processing time belongs to faster or slower processing times of the job (which depends on the machine). The *SL* terminal is included since it is commonly used in some standard DRs, as well as a terminal in many other studies. The remaining two terminals are more machine-centric, with *TMR* allowing us to determine how soon the considered machine will be available, and *TFMA* how soon the machine with the shortest processing time will be available. The latter is useful since jobs will quite often be scheduled on the machine with the fastest processing time. Therefore the inclusion of such a terminal has proven useful.

Apart from the terminal nodes, it is also mandatory to define a set of functional nodes which are used by GP in order to combine the terminal nodes into meaningful expressions. Table 2 represents the set of functional nodes which were used in this paper. These operators were chosen based on the results obtained in a previous study [10]. The basic arithmetic operators were chosen since they denote the minimal set needed to represent basic mathematical expressions. The *POS* node was included since it is also quite often used in certain standard DRs, and since it achieved better results than when using just the absolute value. Other functional nodes, like branching nodes (*ifgt*) or more sophisticated mathematical nodes (*min* and *max*) were also tried out, but unfortunately they did not lead to any significant improvements of the generated DRs. During the evolution process, GP uses both the functional and terminal nodes in order to generate expressions which represent priority functions in the DR. An example of such a priority function is given with the following expression:

$$\pi = pos\left(\frac{pos(w*SL)*(SL+pmin)}{w*pavg}\right) - (pos(w*pt)+(w*age)) \\ - \left(\frac{pmin}{w} - (TMR-age)+(dd-pt)+(pmin+pavg)\right).$$

5 Ensemble learning methods for GP

In this section the four ensemble learning approaches, which were used in order to create ensembles of DRs, will be shortly described. But before the ensemble learning approaches can be described, first it must be defined in which ways the

Table 2 Functional nodes

Node name	Description
$+$	Binary addition operator
$-$	Binary subtraction operator
$*$	Binary multiplication operator
/	Secure binary division: $/(a,b) = \begin{cases} 1, & \text{if } \lvert b\rvert < 0.000001 \\ \frac{a}{b}, & \text{else} \end{cases}$
POS	Unary operator: $POS(a) = max\{a,0\}$

evolved ensembles will be combined into a single DR. For this task two simple ensemble combination methods will be used: sum and vote. The sum combination method will simply sum the priority values of all DRs in the ensemble to get a priority value which will be used to schedule jobs (in the same way as shown in Algorithm 1). On the other hand, the vote method functions a bit differently, as shown in Algorithm 2. For each machine the vote method will first determine which job received the most votes, and then it will choose the one with most votes in a pairwise comparison. Naturally, it is possible that ties occur for both of the ensemble combination methods (although it is more probable that they occur for the vote method). If such a situation occurs, then the job with the earlier release time will be scheduled first.

The implementations of the algorithms described in this section have been done using the evolutionary computation framework (ECF) [20].

Algorithm 2 Vote combination method

```
1: Let $bestPair$ represent the best selected job-machine pair (empty at the beginning)
2: for each machine do
3:     for each DR in the ensemble do
4:         Calculate the priority value by using the selected DR for all jobs
5:         Determine the job for which the DR achieved the best value and vote for it
6:     end for
7:     Select the job with the most votes, or the one which is released earlier if ties occur
8:     Let $currentPair$ denote the job-machine pair chosen in this iteration
9:     for each DR in the ensemble do
10:        Make a vote between $currentPair$ and $bestPair$
11:    end for
12:    if $currentPair$ received more votes than $bestPair$ then
13:        $bestPair := currentPair$
14:    end if
15: end for
16: Schedule the job in the $bestPair$ on the machine in the $bestPair$
```

5.1 Simple ensemble combinations

In this section the simple ensemble combination (SEC) approach will be described. The motivation for this approach comes from the fact that GP is usually executed several times, because of its stochastic nature, in order to obtain good DRs. Thus, it makes sense to see if maybe a combination of the generated DRs could provide better results than a single DR. The idea behind this approach is, therefore, to first evolve several DRs by using a standard GP approach. Following that, an optimal ensemble is determined by trying out various subsets of these DRs, be it by exhaustive search, random search, or some heuristic search method. This approach can be considered similar to some portfolio approaches which combine several metaheuristic methods in order to achieve better results [39, 46].

After obtaining a starting set of DRs by simply repeating the standard GP approach, various subsets of given size will be evaluated as ensembles. In this paper the limit of 20,000 ensemble combinations, which was determined based on initially conducted experiments, will be used. If for the given ensemble size there are less possible combinations of DRs than the limit (i.e. for smaller ensembles), then exhaustive search is used in order to determine the optimal ensemble. Otherwise, 20,000 different ensembles are randomly generated and the one with the best fitness value is chosen. Naturally, there is a possibility that when randomly generating ensembles the same ensemble could be generated several times, but due to the sheer amount of combinations this is very unlikely. Also, creating random combinations of decision makers has previously proven to perform better than ensembles created by using high quality decision makers [17].

The main advantage of this approach is that no new DRs need to be evolved, but rather existing DRs which were evolved beforehand can be used and combined into ensembles. On the other hand, this approach needs an additional problem instance set which must be used in order to determine the set of DRs which form the optimal ensemble. The only parameters of this approach are the size of the ensemble and the ensemble combination method.

5.2 BagGP

BagGP is an ensemble learning approach which applies bagging to GP [19]. This approach evolves DRs in a way that each DR is evolved on a different training set, which is constructed by sampling with repetition from the original training set. The evolved DRs are then combined to form an ensemble. This approach, in addition to the ensemble size and combination method, has an additional parameter which determines the size of the training set used to evolve the DRs. The size of the newly sampled training set can be set to an almost arbitrary value which can be smaller, larger or equal to the original training set size.

5.3 BoostGP

BoostGP is an approach which applies the AdaBoost [14] algorithm in GP [36, 44]. This ensemble learning approach evolves several DRs so that it weights the training set instances in a way that instances that were solved poorly in previous GP runs get a higher importance in the following GP runs, and that newly evolved DRs focus more on solving such problematic instances. Algorithm 3 denotes the BoostGP approach. The approach is mostly the same as the ones denoted in the literature with one notable difference. Since this algorithm is adapted from the regression problem in which the fitness is usually calculated as the difference between the value which was achieved by the individual and the expected value $|f_i - y_i|$, there is a need to adapt it to the case of evolving DRs where there does not exist an explicit expected value which needs to be achieved, but rather a certain criterion is minimised. Nevertheless, since neither of the criteria tested in this paper can have a value lower than zero, the approach is adjusted to treat zero as the expected values for each criteria.

Algorithm 3 BoostGP approach

```
1: Let $P$ be the number of problem instances in the training set and $T$ the ensemble size
2: Initialize the weights for each problem instance as $D_1(k) = 1/P$
3: for t=1..T do
4:     Run GP with the following fitness function: $fitness_t = \sum_{k=1}^{P}(f(individual, x_k) * D_t(k)) * P$ where $f$ represents the original fitness function
5:     The best individual in the GP run is denoted as $ind$
6:     Compute the loss for each example: $L_k = \frac{|f_t(ind,x_k)|}{max_{k=1..m}(f_t(ind,x_k))}$
7:     Compute average loss: $\bar{L} = \sum_{k=1}^{P} L_k * D(k)$
8:     Compute the confidence for the DR: $\beta_t = \frac{\bar{L}}{1-\bar{L}}$
9:     Update the distribution: $D_{t+1}(k) := \frac{D_t(k)*\beta_t^{1-L_i}}{Z_t}$
10: end for
11: The result: $T$ DRs with the appropriate confidences
```

In order to combine the DRs into a single ensemble, four different combination methods are used. The first two methods are the sum and vote methods described previously. The other two methods represent the weighted sum and vote methods which use the confidences obtained for each DR as the weights which will be used to multiply the vote of the DR in the voting method, and the priority value of the DR in the sum method. In addition to the combination method, the second parameter of this approach is the size of the ensemble which needs to be generated.

5.4 Cooperative coevolution

The cooperative coevolution approach is an evolutionary algorithm approach which divides the optimisation problem into several sub-problems which are then solved independently in order to solve the original problem [43]. Each sub-problem is solved by one sub-population in the evolutionary algorithm, and the only interaction between individuals from different sub-populations is when they are combined for evaluation. Naturally, it is not possible to combine one individual with all individuals from the other sub-populations and calculate its fitness for all the combinations, since this would be too time consuming. For that reason, there usually exists a list which contains a representative from each sub-population. An individual is then evaluated in combination with the representative individuals from other sub-populations. This approach can also easily be used in order to evolve ensembles of DRs in a way that each sub-population evolves a single DR which is then combined with DRs from other sub-populations in order to form an ensemble. The ensemble size and ensemble combination approach are the only parameters which need to be defined for this approach.

5.5 Ensemble subset search

In order to additionally improve the performance of the ensemble learning approaches (SEC, BagGP, BoostGP and cooperative coevolution) after the learning process, an additional search can be performed in order to determine the optimal subset of the DRs which form the ensemble. The intuition behind this approach is that the ensemble which is evolved by the ensemble learning approaches does not have to be optimal, and that it is possible to construct a better ensemble by using only a subset of the DRs contained in the original ensemble. This is especially possible when using approaches where the DRs of the ensemble are evolved independently of each other, like in BagGP or BoostGP. In those approaches the DRs forming the ensemble are evolved in independent GP runs, after which they are collected to form the ensemble. Therefore it is possible that the ensemble contains DRs that do not positively contribute to the quality of the ensemble. In order to remedy this, the original ensemble can be modified by removing the unnecessary DRs from the ensemble and consequentially improving the cumulative performance. By reducing the size of the ensemble, the execution speed and interpretability of the ensemble can also be improved.

This approach takes the ensemble evolved by one of the ensemble learning approaches and uses the DRs that formed the original ensemble in order to build ensembles of smaller sizes. Since the largest ensemble evolved in this paper is of size ten, it is possible to try out all ensemble combinations of smaller sizes in a reasonable amount of time, and therefore to determine the optimal ensemble subset. Therefore, if this approach is applied to an ensemble of size ten, all ensemble combinations of sizes between two and nine will be evaluated. Then either the best overall ensemble subset, or the best ensemble subset of a concrete size can be selected. From the description it can be seen that ESS is similar to the SEC approach, with the differences being that it is applied on an existing ensemble of DRs, and that it constructs ensembles of different sizes (smaller than the original ensemble), unlike the SEC approach which creates ensembles of a predefined size only.

With this approach it is possible not only to decrease the ensemble size but also improve its performance. However, for this step an additional problem instance set needs to be defined, on which the optimal combination of DRs, that will form the ensemble subset, can be determined. After the optimal ensemble combination is determined, it is then used to solve unseen scheduling problems.

6 Results

6.1 Benchmark setup and evaluation

In order to be able to evolve and evaluate DRs and ensembles of DRs, an extensive set of 180 scheduling problem instances has been defined. In order to be able to evolve DRs which are applicable on problems of different sizes, problem instances containing 12, 25, 50 or 100 jobs, and 3, 6 or 10 machines have been generated. The detailed procedure of how the problem instances were generated can be found at the project web site [21].

The set of 180 problem instances was divided into three sets, each containing a third of the problem instances. The first set is the training set, which is used by all the approaches in order to evolve the DRs. The second set is the test set, which is used in order to evaluate the effectiveness of the evolved ensembles. The third set, called the validation set, is an additional problem set which is used by some approaches to determine the optimal combination of DRs to form the ensemble. In this paper, the validation set will be used by the SEC approach and ESS. The SEC approach will use the validation set in order to determine the optimal combination of previously evolved DRs, which will form the ensemble. In the ESS the validation set will be used in order to determine the optimal subset of DRs to form the ensemble.

The total fitness of an individual for a certain criterion is calculated as the sum of the criterion values on each of the individual problem instances. Since it was already mentioned that the instances in a single set can have different characteristics, all objective values were additionally normalised in order for them to have similar values on different problem instances. Additionally, in order to obtain statistically significant results, each experiment was run 30 times and the minimum, median and maximum values were calculated based on those 30 runs. In the SEC approach, one run denotes performing 20,000 random combinations of DRs, and choosing the best one of them (evolving DRs by GP is not considered to be a part of a run since they are evolved up front). On the other hand, in one run of BoostGP and BagGP approaches, the underlying GP method is run once for each ensemble element that needs to be generated. Finally, in the cooperative coevolution approach, one run denotes performing one GP run that simultaneously evolves all the elements of the ensemble. The Mann–Whitney statistical test was used in order to determine if a statistically significant difference between two obtained results exists. The results are considered statistically significant if the obtained p value is smaller than 0.05.

The parameters which were used for the standard GP and by the GP which was used by all ensemble learning approaches are shown in Table 3. The table denotes only those parameters which are shared between all the approaches, while the parameters specific to the ensemble learning approaches will be denoted for each experiment individually. The parameters denoted in the table were obtained through an extensive parameter optimisation procedure for the standard GP. An additional

Table 3 Parameters for the GP

Parameter	Value
Population size	1000
Termination criteria	Maximum number of evaluations (80,000)
Selection	Steady state GP using tournament selection
Tournament size	3
Initialization	*Ramped half-and-half*
Mutation probability	0.3
Maximal tree depth	5
Crossover operators	Subtree, uniform, context-preserving, size-fair
Mutation operators	Subtree, Gauss, hoist, node complement, node replacement, permutation, shrink

point which needs to be addressed here is the fact that the ensemble learning approaches will perform *e**80,000 function evaluations, where *e* denotes the size of the ensemble, while the standard GP will perform only 80,000. Although this may seem to lead to an unfair comparison, the standard GP has shown no improvement in the results when increasing the number of evaluations beyond 80,000. Rather, the results were starting to deteriorate, which means that GP started to overfit on the training set.

The next four sections will present the results obtained on the test set for each ensemble learning approach individually, and the influence of the ensemble parameters on them. For the experiments it was chosen to evolve ensembles between sizes two and ten. Larger ensembles were not used since they did not show to improve the results significantly. Additionally, larger ensembles also need more time to perform the decision about what job should be scheduled next, which could be undesirable in dynamic environments, where scheduling decisions need to be performed quickly. In those experiments only the weighted tardiness criterion will be optimised. The baseline values to which all the experiments in the following four sections will be compared to are the ones achieved by the standard GP denoted in Table 4. In the tables the results which are significantly better (tested with the Mann–Whitney statistical test) than those achieved by the standard GP will be underlined, while on the other hand the overall best results for each column will be shown in bold. Additionally, where possible, it was tested if there is a statistical difference between ensembles of sizes two, five and ten when not using ESS, and sizes of two, five and nine when using ESS. The last section compares the best results achieved by all the approaches with the standard GP for the weighted tardiness criterion, and the other three criteria mentioned earlier in the paper.

6.2 Results for SEC

This subsection will present the results achieved for the SEC approach. Table 5 represents the results which were achieved by this approach for the sum and vote combination methods and various ensemble sizes. For each size and ensemble combination method, 20,000 random subsets of 50 DRs evolved by a standard GP approach were tried out, and the best found ensemble was saved. This subset generation procedure was repeated 30 times in order to obtain statistically significant results. Since for the ensemble sizes of two and three the number of possible combinations is less then 20,000, it was possible to perform an exhaustive search and find the best possible ensemble on the validation set. In order to eliminate the need for another problem instance set which would be used by ESS, the same validation set is also used by ESS in order to find the ensemble subset.

Table 4 Weighted tardiness values achieved by the standard GP approach

Min	Med	Max
15.23	15.94	17.59

Table 5 Results for the SEC approach

Ensemble size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	15.17	–	–	15.23	–	–
3	15.59	–	–	15.86	–	–
4	14.92	15.18	15.93	15.20	15.87	16.23
5	**14.84**	**15.12**	15.76	**14.91**	15.81	16.32
6	14.89	15.55	15.89	15.17	15.70	16.04
7	14.94	15.30	15.84	15.14	15.55	**15.92**
8	14.88	15.37	15.86	15.02	15.81	16.43
9	15.18	15.32	**15.70**	15.20	**15.54**	16.06
10	15.10	15.29	16.07	15.15	15.65	16.35

The sum method has achieved better results than the vote method for all ensemble sizes. Both ensemble combination methods obtained the best results for the ensemble size of five (the sum method a value of 14.84, and the vote method a value of 14.91). Furthermore, when compared to the standard GP, it can be seen that the sum combination method achieved statistically better results for all ensemble sizes larger than three, while the vote combination method achieves statistically better results for ensemble sizes larger than five. The sum combination method has shown to additionally improve the median values by 2.5–5.5% and maximum values by around 10% when compared to the standard GP. For the minimum values improvements up to 2.5% were achieved. Based on the results it can be concluded that this approach is much more stable and more likely to achieve better results than standard GP. When comparing ensembles of sizes five and ten it was shown that for the sum method statistically better results were achieved when using ensembles of size five.

Now, ESS will be applied to the ensembles found by the SEC approach. Here we will try out ESS with ensembles of sizes five (for which the SEC approach achieved the best results) and ten (which offers the most subset combinations). Table 6 represents the results achieved for ESS on the ensemble of size five. Here ESS was unable to find subsets which achieve a better minimum value than that of the entire ensemble in all but one occasion for the ESS of size four. Additionally, it can be seen that, except for the vote combination method with ensemble size of two DRs, in all other experiments significantly better results are achieved than those of standard GP. For example, the ensemble of size four attained by the sum combination method achieved an improvement over the standard GP by 5% for the minimum, 4% for the median, and 10% for the maximum value.

Table 7 represents the results achieved by ESS for the ensemble size of ten DRs. For the sum combination method, all results are significantly better than those of the standard GP. On the other hand, for the vote combination method all results were significantly better except for ensemble sizes of two and seven. The improvements on the median values amount to around 2.5–4% for the sum method, and 1–2.5% for the vote method, therefore the sum method has proven to again outperform the vote

Table 6 Results for the SEC approach of size five with ESS

Ensemble subset size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	15.00	15.25	16.57	15.42	16.20	17.31
3	14.88	**15.21**	15.99	15.33	15.69	**16.10**
4	**14.49**	15.25	**15.76**	**15.17**	**15.63**	16.25

combination method. However, ESS was not able to achieve significantly better results than SEC of the same ensemble size. Regarding the ensemble sizes, here it was shown that there is no significant difference between ensembles of sizes two, five and nine for the sum combination method. On the other hand, for the vote combination method it was shown that ensembles of five and nine achieve significantly better results than ensembles of size two.

6.3 Results for BagGP

In this section the results obtained for the BagGP approach will be presented. Apart from testing the influence of the ensemble size and ensemble combination method on the procedure, the influence of the sampled data set size (*bag size*) will also be analysed. The results for this approach are shown in Table 8.

Depending on the bag size the best results for the sum combination method were achieved by ensembles of different sizes. On the other hand, for the vote combination method, the best results are usually achieved by the larger ensembles (from seven to ten) for most of the bag sizes. This is also backed up by the fact that for the sum method there was no statistical difference between ensembles of sizes two, five and ten for any of the bag sizes which were used. On the other hand, when using the vote combination method, it was shown that ensembles of sizes five and ten always achieve significantly better results than ensembles of size two. When

Table 7 Results for the SEC approach of size ten with ESS

Ensemble subset size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	15.17	15.46	15.83	15.23	16.22	16.95
3	**14.86**	15.57	**15.80**	**15.03**	15.69	**15.90**
4	14.89	15.36	**15.80**	15.27	**15.58**	16.15
5	15.06	15.34	16.00	15.07	15.73	16.17
6	15.08	15.30	16.00	**15.03**	15.75	16.34
7	15.14	**15.28**	15.98	15.14	15.80	16.24
8	15.10	15.29	15.93	15.12	15.71	16.16
9	15.13	15.30	16.07	15.21	15.70	16.28

Table 8 Results for the BagGP approach

Ensemble size	Bag size																	
	30			40			50			60			70			80		
	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max
Sum ensemble construction																		
1	15.42	16.62	18.91	15.39	16.62	20.25	15.12	16.45	18.05	15.25	16.42	22.05	15.16	16.56	21.89	15.42	16.24	18.95
2	15.18	16.41	18.71	15.10	**16.08**	18.97	15.29	16.28	18.05	15.12	16.06	17.09	**14.96**	16.29	20.82	15.02	16.17	18.44
3	15.18	16.43	19.43	14.91	16.35	21.14	15.19	**16.09**	20.31	15.40	16.01	17.09	15.04	16.15	18.85	15.00	16.02	18.23
4	15.59	16.35	18.12	15.27	16.20	21.39	**15.03**	16.35	**17.90**	15.35	15.87	**16.84**	15.13	16.05	20.15	15.00	15.97	17.31
5	15.18	16.30	17.88	15.30	16.32	19.70	**15.03**	16.33	**17.90**	15.38	**15.84**	18.48	15.21	16.01	19.09	14.93	15.97	17.97
6	15.26	16.30	18.90	15.01	16.11	22.72	15.23	16.26	18.66	15.03	16.10	18.48	15.34	16.03	19.33	**14.89**	15.85	17.31
7	15.23	16.40	18.56	15.31	16.10	21.38	15.24	16.41	21.17	15.03	16.11	18.48	15.16	16.02	**17.96**	**14.89**	15.92	17.31
8	**15.01**	16.17	**17.63**	15.31	16.11	**17.77**	15.24	16.30	23.70	**15.00**	15.99	19.04	15.17	**16.00**	18.02	14.91	15.81	**17.24**
9	15.17	16.17	18.20	**14.77**	16.15	17.84	15.20	16.29	23.70	15.06	16.08	19.04	15.36	16.05	22.95	14.91	**15.78**	17.26
10	15.16	**16.10**	19.95	14.78	16.12	17.84	15.12	**16.09**	23.70	15.07	16.13	19.06	15.11	16.08	21.41	15.01	15.87	17.26
Vote ensemble construction																		
1	15.42	16.62	18.91	15.39	16.62	20.25	15.12	16.45	18.15	15.25	16.42	22.05	15.16	16.56	21.89	15.42	16.24	18.95
2	15.31	16.41	19.17	15.13	16.79	21.15	15.56	16.23	17.70	15.28	16.18	24.96	15.13	16.34	21.55	15.34	16.31	20.41
3	15.17	15.85	17.58	14.97	15.79	17.30	15.06	15.89	16.93	**15.05**	**15.55**	16.62	15.31	15.82	16.86	15.09	15.61	16.52
4	15.18	16.04	17.15	15.12	16.01	16.71	15.16	15.89	16.93	15.16	15.81	16.39	15.07	15.81	16.73	15.26	15.74	17.03
5	15.33	15.81	16.94	15.27	15.80	16.62	15.18	15.74	16.54	15.22	15.79	16.61	14.95	15.74	16.38	15.23	15.73	16.50
6	15.02	15.81	16.89	15.19	15.83	17.18	15.08	15.71	16.69	15.24	15.74	17.00	15.05	15.67	16.38	15.18	15.58	16.43
7	15.09	15.62	16.67	15.08	15.72	16.54	15.26	15.59	16.31	15.13	15.61	16.54	15.10	15.60	**16.05**	**15.08**	15.50	16.36
8	15.08	15.83	**16.17**	15.04	15.75	**16.31**	**15.01**	15.61	16.47	15.24	15.69	16.47	15.15	15.65	16.14	15.13	15.46	16.45
9	15.20	**15.61**	16.73	15.27	15.77	16.51	15.12	15.58	16.58	15.09	15.61	**16.46**	**14.88**	**15.59**	16.24	15.14	15.52	**16.16**
10	**15.01**	15.65	16.56	**14.95**	**15.63**	16.42	15.04	**15.48**	16.15	15.15	15.66	16.47	14.93	15.61	16.21	15.11	**15.40**	16.32

comparing ensembles of sizes five and ten, it was shown that for bag sizes of 30, 50 and 80, ensembles of size ten achieved significantly better results. The overall best result achieved by the sum method was for the bag size of 40 instances and ensemble of size nine (a value of 14.77), while for the vote method the best achieved result was for the bag size of 70 instances and ensemble of size nine (a value of 14.88).

By comparing the two ensemble combination methods with each other, it can be seen that the sum combination method achieves better minimum values for bag sizes of 40 and 80, while for the other sizes the minimum values the two methods achieve similar values. On the other hand, the median values of the evolved ensembles are consistently better when using the vote combination method. Based on the minimum and maximum values the methods achieved, it is evident that the vote combination method achieves less dispersed results, and is thus more stable. When compared to the standard GP, the ensembles were in most cases able to either find solutions which were better or of the same quality as the best solution found by the standard GP. Nevertheless, there were a few experiments in which the best found solution by BagGP was worse then the best found solution from the standard GP (usually for a smaller ensemble size). Regarding the median values, the sum method usually achieved values which were worse than the median value achieved by the standard GP, while the vote method usually achieved better ones or values comparable to the one achieved by standard GP. The statistical test shows that the sum method did not achieve even one result which is better than the standard GP. However, the vote combination method consistently achieved significantly better results for larger ensemble sizes (of size seven and larger). For the vote combination method, the BagGP approach was able to achieve improvements over the standard GP for the minimum value by 2.4% and median by 3.3%. Out of the results which are significantly better than those of the standard GP, the biggest improvements for the median values were achieved when using the bag size of 50 and 80 problem instances, in both cases for the ensemble of size nine.

The standard BagGP approach will additionally be enhanced with ESS described earlier. It was chosen to apply ESS only on the ensemble of size ten. Since the number of subsets of ten DRs is not large, an exhaustive search was performed in order to be able to find the optimal ensemble subset for each ensemble obtained from the 30 runs of the BagGP approach. The results obtained for this enhancement are shown in Table 9. The vote method achieves a better median value on the 30 runs for larger ensemble sizes, while for smaller ensemble sizes the sum combination method achieves a better median value. Additionally, the vote method has also shown to be more stable than the sum method, which can be seen from the fact the difference between the minimum and maximum values is smaller when using the sum combination method. The best result achieved by the sum method was when using the bag size of 30 problem instances and ensemble subset size of four (a value of 14.41). On the other hand the best result for the vote method was achieved in two situations, one for the ensemble subset size of four and the bag size of 40 problem instances, and the other for the ensemble subset size 9 and bag size 70 problem instances (a value of 14.85). By comparing the results with the standard GP it can be noticed that the sum combination method now achieves significantly better

Table 9 Results for the BagGP approach with ESS

Ensemble subset size	Bag size																	
	30			40			50			60			70			80		
	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max	Min	Med	Max
Sum ensemble construction																		
2	15.20	15.97	17.35	15.10	15.93	18.58	15.33	16.00	17.23	15.30	15.75	17.01	15.12	15.82	**16.64**	15.09	15.83	17.19
3	14.86	16.01	**17.02**	**14.94**	**15.72**	**16.90**	15.41	16.22	17.10	**15.02**	15.62	16.73	15.35	15.82	16.74	15.03	15.67	16.27
4	**14.41**	16.06	17.54	15.41	15.89	17.79	**14.66**	15.92	**16.88**	15.06	15.70	16.64	15.15	15.66	16.67	**14.81**	**15.56**	17.07
5	14.53	**15.93**	17.61	15.31	15.89	17.81	14.87	**15.72**	**16.88**	15.06	**15.61**	**16.48**	**14.98**	15.76	16.97	15.05	15.77	17.15
6	15.19	16.05	17.41	14.96	15.95	17.89	15.09	15.95	17.69	15.06	15.62	16.52	15.03	**15.64**	18.11	14.89	15.80	17.15
7	14.97	16.11	17.18	15.07	16.03	17.87	14.84	15.95	17.53	15.06	15.70	17.18	15.25	15.67	18.34	14.92	15.76	17.19
8	15.09	16.11	17.54	15.19	16.14	17.65	15.09	16.06	17.01	15.16	15.76	16.82	15.25	15.84	17.59	15.02	15.78	17.30
9	15.04	16.15	19.66	15.17	16.10	17.73	15.12	15.90	17.04	15.05	16.00	17.25	15.33	15.90	18.22	15.16	15.80	17.26
Vote ensemble construction																		
2	15.33	16.10	17.83	14.99	16.16	17.60	15.12	16.14	17.86	15.02	15.95	16.98	15.12	15.92	17.29	15.04	15.95	17.73
3	15.10	15.80	16.98	15.10	15.84	16.61	15.04	15.65	16.87	14.96	15.77	16.71	15.01	15.85	16.40	15.14	15.60	16.56
4	15.26	15.93	16.69	**14.85**	15.85	16.73	15.01	15.74	16.78	15.12	15.82	16.65	15.04	**15.50**	16.41	15.07	15.64	16.31
5	15.08	**15.63**	16.81	15.03	15.70	16.72	14.90	**15.54**	16.57	15.09	15.66	16.37	15.01	15.65	16.29	15.05	**15.41**	16.36
6	15.11	15.82	16.56	15.05	15.73	16.76	15.00	15.65	16.59	15.10	15.68	16.42	15.02	15.59	16.14	**14.94**	15.52	16.50
7	**14.99**	15.72	16.64	15.05	**15.62**	16.72	**14.86**	15.63	**16.38**	15.09	15.55	16.49	14.93	15.61	**16.03**	14.99	15.51	**16.10**
8	15.19	15.67	**16.47**	15.17	15.72	16.68	15.07	15.61	16.45	15.07	**15.53**	16.47	15.03	15.65	16.14	15.03	15.47	16.21
9	15.21	15.67	16.73	15.00	15.67	**16.65**	15.00	15.56	16.51	**15.06**	15.60	**16.31**	**14.85**	15.57	16.10	15.15	15.43	16.16

results for larger bag sizes and smaller to medium ensemble sizes. In addition to achieving significantly better results for larger ensemble sizes, the vote combination method now also achieves significantly better results even for smaller and medium sized ensembles, when using larger bag sizes. The improvements over the standard GP for the vote method are mostly the same as without using ESS. The sum method achieved improvements up to 2.8% for the minimum value and 3.4% for the median. The largest improvements for the median value were achieved by using the bag size of 80, for both combination methods.

By comparing the results of ESS and the results achieved without it, it was shown that, when using the sum combination method, in 70% of experiments ESS was able to achieve significantly better results than the original ensemble which was generated by BagGP. On the other hand, for the vote combination method, the number of significantly better results achieved by ESS was only 24%. Both methods achieved the best overall result when additionally using ESS. Based on all the previously outlined points it is possible to conclude that ESS should be used with the sum method since it can lead to significantly better results, but can also be used with the vote method since it also leads to improvements in the results.

6.4 Results for BoostGP

This section presents the results obtained for the BoostGP approach. Table 10 represents the results achieved by the unweighted BoostGP approach. The vote combination method achieves better minimum and median values for all ensemble sizes except for size two. When using the sum combination method there was no significant difference for the results achieved between ensembles of sizes two, five and ten. On the other hand, for the vote method it was shown that ensembles of sizes five and ten achieved significantly better results than the ensembles of size two. For the sum method the best result was achieved for the ensemble of size nine (a value of 15.09), while the sum method achieved the best result for the ensemble of size five (a value of 14.93). By comparing the results with the standard GP it can be seen

Table 10 Results for the unweighted BoostGP approach

Ensemble size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
1	15.12	16.00	18.13	15.12	16.00	18.13
2	15.26	15.82	17.33	15.27	16.02	18.01
3	15.28	**15.74**	17.45	15.02	15.72	16.42
4	15.22	15.86	17.45	15.09	15.67	16.52
5	15.37	15.89	17.52	**14.93**	**15.50**	16.11
6	15.16	15.86	17.81	15.08	15.65	16.14
7	15.14	15.84	**16.42**	14.99	15.59	**16.00**
8	15.14	15.86	**16.42**	15.15	15.62	16.05
9	**15.09**	15.86	16.76	15.10	15.52	16.49
10	15.14	15.85	16.78	15.08	15.54	16.24

that the sum combination method was not able to achieve significantly better results, while the vote combination method achieved significantly better results in all cases except for the two smallest ensemble sizes. The improvements which the vote method can achieve over the standard GP reach up to 2% for the minimum and 2.8% for the median value, which were obtained by using an ensemble of size five.

After obtaining results for the BoostGP approach, ESS is used in order to find optimal subsets of DRs to form an ensemble. These results are shown in Table 11. The vote method achieved better median values for larger ensembles, while the sum method for smaller ensembles. Between ensemble sizes of two, five and nine there was no significant difference for the sum combination method. For the vote method, as previously, ensembles of sizes five and nine achieved significantly better results than ensembles of size two. The sum method was able to achieve significantly better results than standard GP for ensemble sizes of three and four, while the vote method achieved significantly better results for all ensemble sizes except for the smallest. The largest improvements, for both methods, over the standard GP were around 3% for the minimum value and 2.8% for the median. When compared to the results achieved by BoostGP, ESS was able to significantly improve the results only for the sum method with ensemble sizes of four, five and six.

Table 12 represents the results obtained when using confidences as weights in the sum and vote combination methods. Once again the vote method achieves better median values for larger values, while the sum method achieves better results for smaller ensembles. By comparing these results to the ones achieved by the unweighted approach, it can be seen that the overall best solution was achieved when using the weighted approach. For the sum combination method there was no significant difference between the results achieved by ensembles of sizes two, five and ten. On the other hand, for the vote method ensembles of size ten achieved significantly better results than ensembles of size five. The weighted BoostGP approach achieved significantly better results than standard GP when using the vote combination method and for ensemble sizes larger than three. The achieved

Table 11 Results for the unweighted BoostGP approach with ESS

Ensemble subset size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	15.09	15.83	17.19	15.10	16.04	16.82
3	15.03	15.67	16.27	15.02	15.67	16.63
4	**14.81**	**15.56**	17.07	15.15	15.67	16.19
5	15.05	15.77	17.15	**14.78**	15.63	16.13
6	14.89	15.80	17.15	14.93	15.66	16.25
7	14.92	15.76	17.19	14.98	15.60	16.34
8	15.20	15.78	17.30	15.03	15.62	16.49
9	15.16	15.80	17.26	14.99	**15.54**	**16.11**

Table 12 Results for the weighted BoostGP approach

Ensemble size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
1	15.12	16.00	18.13	15.12	16.00	18.13
2	15.39	15.85	17.01	15.16	15.93	18.28
3	15.30	15.77	17.91	15.10	15.90	17.93
4	15.16	15.91	17.45	**14.83**	15.61	16.47
5	15.05	15.90	17.61	14.87	15.66	16.48
6	15.05	15.96	17.63	15.14	15.71	**16.30**
7	**14.95**	**15.79**	**16.42**	14.96	15.56	16.35
8	15.19	15.81	16.54	15.09	15.57	16.66
9	15.05	15.89	16.54	15.02	**15.53**	16.50
10	15.15	15.82	16.54	14.96	15.54	16.45

improvements over the standard GP are roughly the same as without using the weights.

Finally, the weighted BoostGP approach will be used with ESS to further improve the results. Table 13 represents the results achieved for the weighted BoostGP approach by additionally using ESS. With ESS, BoostGP has achieved significantly better results in six out of eight experiments for both the sum and vote methods. The improvements over the standard GP are the same when using ESS without the weights. ESS was able to significantly improve results for the sum method when using ensmbles of sizes three, four and five. For the vote method it was only able to improve the result for the ensemble of size three.

6.5 Results for cooperative coevolution

In this section the results obtained by the cooperative coevolution approach will be presented. For this approach two configurations depending on the termination criterion are used. The first configuration uses a termination criterion which depends on the number of DRs that are evolved for the ensemble, namely

Table 13 Results for the weighted BoostGP approach with ESS

Ensemble size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	**14.77**	15.75	16.66	15.10	16.01	16.63
3	15.13	15.71	17.00	15.11	15.73	16.14
4	14.92	**15.66**	16.43	15.09	15.82	16.71
5	14.94	15.74	16.80	14.99	15.63	**16.01**
6	15.05	15.71	16.59	14.97	15.52	16.38
7	15.02	15.71	16.59	15.10	15.58	16.61
8	15.02	15.72	17.31	15.02	**15.44**	16.29
9	14.97	15.79	**16.42**	**14.94**	15.62	16.50

80,000*e evaluations, where e represents the number of DRs in the ensemble. On the other hand, the second configuration will use 80,000 evaluations. Thus, with larger ensembles, the GP will have less iterations on its disposal to evolve a good solution. The idea behind the second configuration is to speed up the entire procedure, since the first configuration can take quite some time to evolve the ensembles, especially for those of larger sizes.

Table 14 represents the results obtained by the first configuration. Immediately it can be seen that only the sum method, with the ensemble size of two DRs, was able to achieve a better solution than the standard DR. For the sum method, it was observed that the quality of ensembles deteriorates as the number of DRs in them increases, thus the best results are achieved for smaller ensembles. This is backed up with statistical tests which show that ensembles of size two achieve significantly better results than ensembles of sizes five and ten, and also that ensembles of size five achieve significantly better results than ensembles of size ten. For the vote method the situation is the same, with ensembles of smaller sizes achieving significantly better results.

In no occasion did this method achieve significantly better results when compared to the standard GP. Through statistical tests it was shown that when using the sum method with ensembles of sizes two and three, there was no significant difference between the standard GP and the cooperative coevolution approach. However, in all other experiments it was shown that the cooperative coevolution achieved significantly worse results than the standard GP.

The results obtained by the second configuration, which uses only 80,000 evaluations, are shown in Table 15. The overall best solution was obtained by the vote method for the ensemble size of three DRs. For both methods it was shown that ensembles of size two achieved significantly better results than those of ensemble sizes five and ten. Once again neither of the experiments achieved better results than the standard GP. The statistical tests have again shown that the achieved results are worse than those achieved by the standard GP.

By comparing the two configurations with each other, it is possible to determine the influence of the termination criterion on the results of the cooperative

Table 14 Results for the coevolution approach with the first configuration

Ensemble size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	**15.11**	**15.98**	**16.43**	15.45	**16.25**	**16.95**
3	15.35	15.99	16.95	15.44	16.37	18.37
4	15.64	16.54	18.16	15.59	16.81	20.57
5	15.61	16.60	18.53	15.47	16.71	18.48
6	15.72	17.10	18.31	**15.39**	17.11	25.49
7	15.52	16.68	17.86	15.52	17.54	20.01
8	16.38	17.35	20.92	16.00	16.88	20.62
9	15.77	18.33	23.26	15.50	17.01	18.77
10	16.51	17.52	21.27	15.56	17.26	19.49

Table 15 Results for the coevolution approach with the second configuration

Ensemble size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	15.21	**16.09**	23.39	15.33	16.30	17.53
3	**15.18**	16.43	**18.80**	**15.14**	**15.97**	**16.96**
4	15.49	16.73	22.57	15.23	16.27	18.50
5	15.83	17.35	23.39	15.76	16.49	18.79
6	15.67	16.78	23.68	15.30	16.60	19.26
7	15.37	17.19	22.55	15.70	16.64	19.79
8	15.83	17.13	25.74	15.62	17.30	18.93
9	15.50	17.33	26.23	16.24	17.59	20.18
10	16.13	17.57	19.66	15.81	16.90	17.86

coevolution approach. It can be seen that neither configuration has consistently achieved better results over the other. Therefore, the first configuration does not offer any advantage over the second, even though it uses a larger number of evaluations.

Finally, Table 16 represents the results obtained for the first configuration by using ESS on the ensemble of size ten. ESS was unable to improve the results which can be seen from the fact that none of the experiments were able to achieve significantly better results than standard GP. In addition to that, ESS was also unable to significantly improve the results when compared to the original results of cooperative coevolution.

Table 17 represents the results for applying ESS on the second configuration. For both methods it was shown that significantly better results were achieved by ensembles of sizes five and nine when compared to ensembles of size two. As in the previous case, ESS was unable to improve the results over the standard GP or the cooperative coevolution method.

Table 16 Results for the coevolution approach with ESS (first configuration)

Ensemble subset size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	16.52	20.41	33.34	16.62	18.72	24.15
3	15.77	18.10	22.66	16.24	17.91	23.39
4	15.75	17.56	19.91	**15.23**	17.64	22.18
5	15.78	17.39	21.04	16.14	17.39	21.20
6	15.74	17.52	**19.90**	15.51	17.14	22.46
7	15.75	**16.77**	20.69	15.80	**16.92**	**19.23**
8	15.75	17.47	20.68	15.57	17.01	19.60
9	**14.98**	17.31	21.74	15.58	17.12	21.49

6.6 Comparison of ensemble learning approaches

This section will compare the performances between the different ensemble learning approaches on four scheduling criteria. For the Twt criterion the results which were displayed in the last four subsections will be aggregated. On the other hand, the results for the other three criteria were not as fine tuned as for the Twt criterion. Therefore better results could very likely be achieved if the parameters were further optimised for each given criterion.

Before analysing the results, the nomenclature of the approaches must first be described. The number alongside each approach will denote the size of the ensemble which was used. The ESS flag denotes that ESS was used to find a subset of ensembles and the size of the subset is denoted alongside the flag. The B flag in the BagGP approach denotes the bag size of problem instances which was used for evolving ensembles. This flag will only be denoted for the Twt criterion, since for the other criteria a bag size of 40 was used for all experiments. The C flag denotes that the confidences are used as weights in the BoostGP approach. Finally, the con1 and con2 flags denote that the first or the second configuration is used with the coevolution approach. Additionally, this table includes a column denoted with *p*, which represents the *p* value obtained by the Mann–Whitney statistical test. The tests were performed in order to test whether there is a statistical difference between the ensemble learning approaches and the results obtained by the standard DRs. Some values are denoted with "<0.001" which means that a value less than 0.001 was obtained for the *p* value.

First the approaches will be compared by using the Twt criterion. Table 18 represents the best results from the tested approaches, which were aggregated from the previous four subsections. With the optimised parameters, each one of the tested ensemble learning approaches was able to find a better solution than the standard GP. The best result, with a value of 14.41, was achieved by the BagGP approach with the addition of ESS. This represents an improvement of 5.4% over the best result achieved by the standard GP. The best results were generally achieved by the

Table 17 Results for the coevolution approach with ESS (second configuration)

Ensemble subset size	Sum			Vote		
	Min	Med	Max	Min	Med	Max
2	**15.80**	19.87	34.80	15.60	18.17	22.80
3	15.86	18.16	28.44	16.28	17.33	19.62
4	15.86	17.90	21.31	15.76	17.29	20.34
5	15.86	**17.31**	20.54	15.50	16.89	19.64
6	15.81	17.57	20.54	**15.39**	**16.75**	**18.55**
7	16.10	17.59	20.28	15.75	**16.75**	19.11
8	16.10	17.71	**19.52**	15.43	16.76	19.46
9	16.10	17.58	20.71	15.50	16.79	19.29

BagGP and SEC approaches, while the BoostGP also achieved good results, but not to such a great extent. The greatest improvement over the standard GP in the median value, amounting to 5.4%, was achieved by the SEC approach. The cooperative coevolution procedure has achieved the worst results of all the ensemble learning approaches. Figure 1 represents a box plot representation of the achieved results (in order to stand out, the results for standard GP have been coloured with a specific pattern). The box plot outlines several interesting characteristics of the tested approaches for this criterion. First of all, the SEC approach with the sum combination method achieves solutions which are least dispersed among all the approaches. The solutions found by BagGP are largely dispersed when using the sum combination method, but by using the vote combination method the dispersion is reduced. Additionally, for the vote combination method the solutions of most ensemble learning approaches tend to be less dispersed than when using the sum combination method. Regarding the statistical difference between the ensemble learning approaches and the standard DRs, it can be seen that in most cases there is a statistically significant difference (which is especially evident when using the vote combination method). It is interesting to note that although the BagGP-10 ESS-4 B30 approach achieved the single best solution, there seems to be no significant difference when comparing all the results achieved by this approach with the results achieved by the standard GP.

Table 18 Result comparison for the Twt criterion

Approach	Min	Med	Max	*p*
Standard GP	15.23	15.94	17.59	–
Sum ensemble construction				
SEC-5	14.84	15.12	15.76	<0.001
SEC-5 ESS-4	14.49	15.25	15.76	0.001
BagGP-9 B40	14.77	16.15	17.84	0.040
BagGP-10 ESS-4 B30	14.41	16.06	17.54	0.183
BoostGP-9	15.09	15.89	16.76	0.669
BoostGP-10 ESS-4	14.81	15.56	17.07	0.015
BoostGP-7 C	14.95	15.79	16.42	0.279
BoostGP-10 C ESS-2	14.77	15.75	16.66	0.026
Coevolution-2 con1	15.11	15.98	16.43	0.669
Vote ensemble construction				
SEC-5	14.91	15.81	16.32	0.024
SEC-10 ESS-3	15.03	15.69	15.90	<0.001
BagGP-9 B70	14.88	15.59	16.24	<0.001
BagGP-10 ESS-4 B40	14.85	15.85	16.73	0.330
BoostGP-5	14.93	15.50	16.11	<0.001
BoostGP-10 ESS-5	14.78	15.63	16.13	<0.001
BoostGP-4 C	14.83	15.61	16.47	0.005
BoostGP-10 C ESS-9	14.94	15.62	16.50	0.001
Coevolution-3 con2	15.14	15.97	16.96	0.355

Fig. 1 Box plot representation of results for the Twt criterion

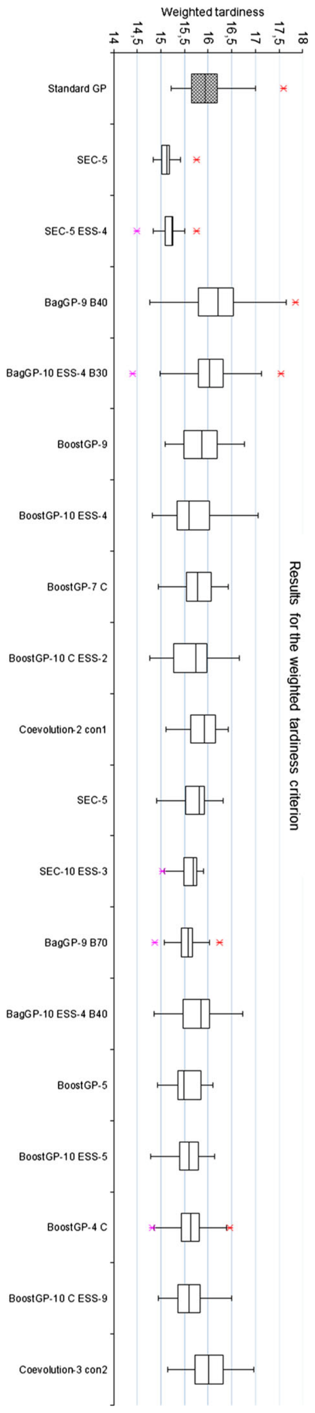


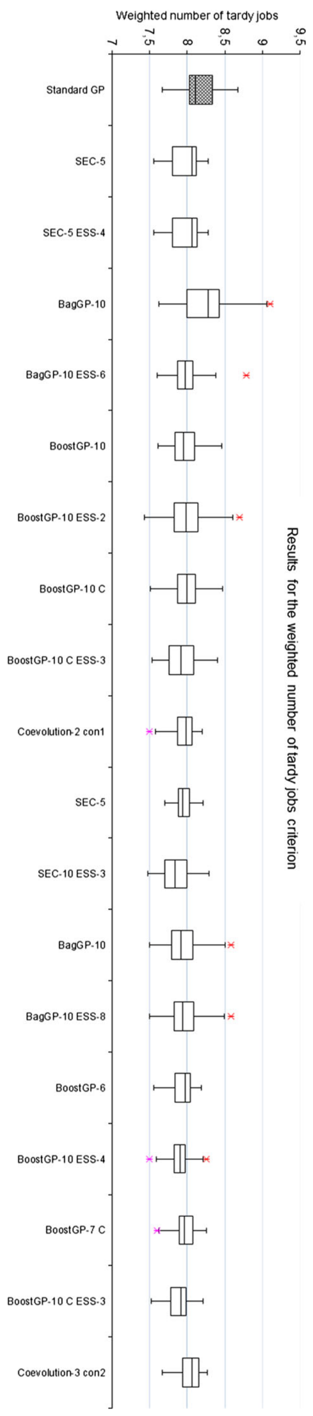
Table 19 represents the results the ensemble approaches achieved for the Nwt criterion. Here the best result, with a value of 7.435, was achieved by the BoostGP approach with ESS. When compared to the best result achieved by the standard GP, this represents an improvement of 3.1%. Even though an exhaustive parameter optimisation was not performed for this criterion, most of the ensemble learning approaches were able to outperform the standard GP, to a larger or smaller extent. The Coevolution-2 approach with the sum combination method achieved one of the better results among all the approaches, unlike for the Twt criterion where it did not achieve significantly better results than the standard GP. Figure 2 represents the box plot representation of the results achieved for the Nwt criterion. Out of all the approaches, the SEC approach with the vote combination method achieved the smallest dispersion out of the results, while the BagGP has achieved the largest dispersion of the results. It is also noticeable that the ensemble learning approaches achieved less dispersed solutions when using the vote combination method. From the table it is also interesting to note that only in one occasion, for the BagGP-10 approach, there is no significant difference between the results obtained by the ensemble methods and by the standard DRs.

Table 20 represents the results achieved for the Ft criterion. Here the best result of 157.1 was achieved by two approaches, and it represents an improvement of around 0.7% when compared to the best result of the standard GP. The BagGP and

Table 19 Result comparison for the Nwt criterion

Approach	Min	Med	Max	*p*
Standard GP	7.674	8.107	8.669	–
Sum ensemble construction				
SEC-5	7.556	8.064	8.276	0.008
SEC-5 ESS-4	7.556	8.064	8.276	0.016
BagGP-10	7.621	8.284	9.105	0.210
BagGP-10 ESS-6	7.601	8.001	8.827	0.002
BoostGP-10	7.616	7.949	8.455	0.001
BoostGP-10 ESS-2	7.435	7.989	8.694	0.003
BoostGP-10 C	7.516	7.995	8.472	0.005
BoostGP-10 C ESS-3	7.536	7.921	8.399	<0.001
Coevolution-2 con1	7.505	7.980	8.196	<0.001
Vote ensemble construction				
SEC-5	7.699	7.946	8.212	<0.001
SEC-5 ESS-3	7.476	7.834	8.287	<0.001
BagGP-10	7.496	7.917	8.582	<0.001
BagGP-10 ESS-8	7.505	7.946	8.579	<0.001
BoostGP-6	7.560	7.979	8.195	<0.001
BoostGP ESS-4	7.497	7.912	8.257	<0.001
BoostGP-7 C	7.606	7.962	8.253	<0.001
BoostGP-10 C ESS-3	7.528	7.918	8.215	<0.001
Coevolution-3 con2	7.671	8.062	8.272	0.023

Fig. 2 Box plot representation of results for the Nwt criterion



BoostGP approaches were again those which achieved the best results, while the cooperative coevolution approach has achieved the worst results, even worse than those of the standard GP for the vote combination method. Figure 3 represents the box plot representation of the results achieved for the Ft criterion. This time the least dispersed solutions are achieved by using the BoostGP approach with the vote combination method, while the most dispersed solutions were achieved by the BagGP approach. Even for this criterion it can be seen that less dispersed solutions are mostly achieved when using the vote combination method. By examining the statistical difference between the ensemble learning approaches and the standard DRs, it can be seen that there is a certain amount of cases in which there is no significant difference between those approaches. However, such a result is expected since the improvement by the ensemble learning approaches for this criterion was not that significant.

Table 21 represents the results achieved for the last criterion, the makespan. The best result was achieved by the BoostGP which uses confidences as weights. The criterion value achieved by this approach is 38.20, which represents an improvement of only 0.3% in comparison to the best result achieved by the standard GP. For this criterion most ensemble learning approaches were struggling to outperform the best result achieved by the standard GP, with some approaches even achieving worse results when compared to the standard GP. Figure 4 shows the box plot

Table 20 Result comparison for the Ft criterion

Approach	Min	Med	Max	p
Standard GP	158.1	159.3	161.6	–
Sum ensemble construction				
SEC-5	157.6	158.7	160.3	<0.001
SEC-5 ESS-4	157.1	158.6	159.8	<0.001
BagGP-9	158.1	160.0	163.5	<0.001
BagGP-10 ESS-5	157.1	159.2	162.1	0.261
BoostGP-3	157.8	159.6	161.7	0.335
BoostGP-10 ESS-7	157.2	159.4	160.6	0.777
BoostGP-5 C	157.5	159.5	161.7	0.585
BoostGP-10 C ESS-6	157.3	159.1	160.1	0.183
Coevolution-2 con1	158.1	159.4	160.3	0.842
Vote ensemble construction				
SEC-5	157.6	158.5	159.4	<0.001
SEC-5 ESS-3	157.8	158.8	159.7	0.001
BagGP-5	158.0	159.7	161.1	0.192
BagGP-10 ESS-2	157.5	159.8	161.4	0.032
BoostGP-9	157.7	158.8	159.8	<0.001
BoostGP-10 ESS-8	157.2	158.8	159.9	<0.001
BoostGP-6 C	157.5	158.7	160.3	0.002
BoostGP-10 C ESS-2	157.1	159.3	160.7	0.355
Coevolution-3 con2	158.4	160.0	161.6	0.003

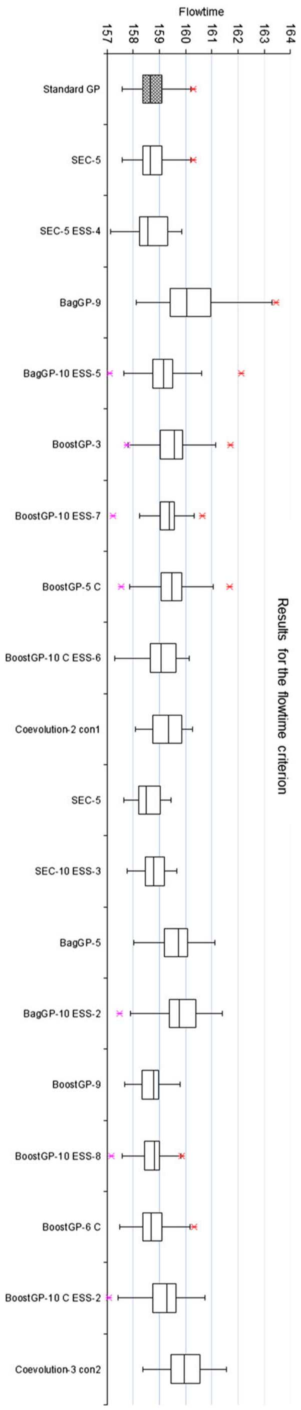


Fig. 3 Box plot representation of results for the Ft criterion

representation of the results achieved for the Cmax criterion. The SEC approach once again achieves the least dispersed solutions. On the other hand, BagGP has again shown to achieve the most dispersed solutions. Even for this criterion the approach achieved less dispersed solution when the vote combination method was used. The statistical tests have shown that in several cases there was no significant difference between the results obtained by the ensemble learning approaches and the results obtained by the standard GP (which is especially evident when using the sum combination method), but as with the Ft criterion such a behaviour is expected because of the smaller improvements achieved for this criterion.

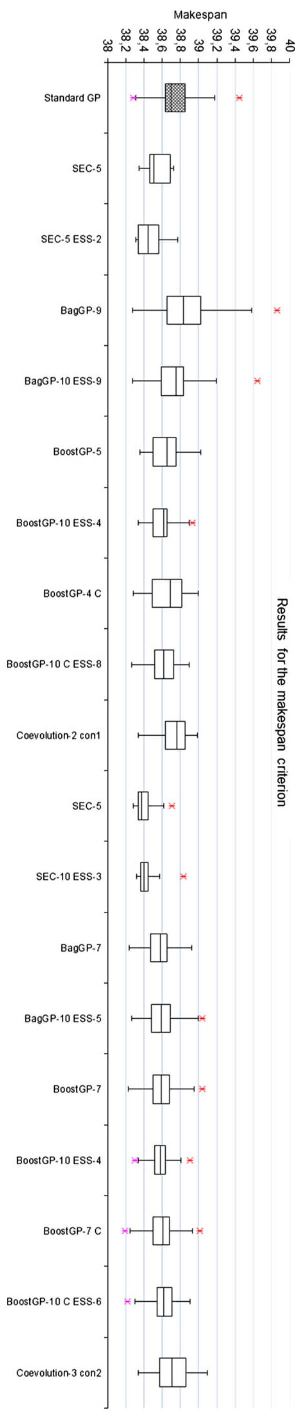
7 Discussion

This section gives a short discussion on the results achieved by all the tested ensemble learning approaches. First, each approach will be discussed individually, after which all the approaches will be compared with each another.

Table 21 Result comparison for the Cmax criterion

Approach	Min	Med	Max	*p*
Standard GP	38.29	38.70	39.45	–
Sum ensemble construction				
SEC-5	38.34	38.51	38.73	<0.001
SEC-5 ESS-2	38.31	38.45	38.77	<0.001
BagGP-9	38.27	38.83	39.86	0.101
BagGP-10 ESS-9	38.27	38.75	39.64	0.874
BoostGP-5	38.35	38.65	39.02	0.054
BoostGP-10 ESS-4	38.33	38.62	38.93	<0.001
BoostGP-4 C	38.28	38.69	38.99	0.126
BoostGP-10 C ESS-8	38.27	38.62	38.90	0.003
Coevolution-2 con1	38.34	38.76	38.99	0.648
Vote ensemble construction				
SEC-5	38.28	38.37	38.71	<0.001
SEC-5 ESS-3	38.22	38.40	38.83	<0.001
BagGP-7	38.24	38.58	38.92	<0.001
BagGP-10 ESS-5	38.27	38.59	39.04	<0.001
BoostGP-7	38.23	38.59	39.04	<0.001
BoostGP-10 ESS-4	38.30	38.58	38.90	<0.001
BoostGP-7 C	38.20	38.61	39.02	<0.001
BoostGP-10 C ESS-6	38.22	38.61	38.91	0.008
Coevolution-3 con2	38.33	38.71	39.10	0.713

Fig. 4 Box plot representation of results for the Cmax criterion



7.1 SEC

Although it is quite a simple approach, the SEC approach has nevertheless shown to be able to outperform the best result from the standard GP in most of the experiments. Regarding the combination method, we can conclude that the sum combination method is more effective since it consistently achieved better results. Also it was shown that the SEC approach achieves the best results when using an ensemble of moderate size (from around four to eight DRs). This is probably a consequence of the fact that smaller ensembles do not have the expressive value of the medium and larger sized ensembles, while on the other hand it is much more difficult to find a good ensemble combination for larger ensemble sizes because of the random choice of DRs which form the ensemble.

The most obvious benefit of this approach is that it can be used with already existing DRs, and thus it can eliminate the need for evolving new rules. But, even if new DRs need to be evolved, the time needed for that is significantly smaller then when evolving entire ensembles by some of the other three approaches. One additional benefit of this approach, which became evident after analysing the results, is that it achieves solutions with results which are less dispersed than the results achieved by other approaches. This means that the approach has a higher probability of achieving good solutions. Therefore the second benefit of this approach is its speed and the possibility to quickly create ensembles of DRs which are not greatly dispersed. The main drawback is that it requires an additional problem set on which it will determine the DRs that form the ensemble, since using the same set which was used for evolving DRs could possibly lead to overfitting.

7.2 BagGP

The results achieved for the BagGP offer some interesting conclusions about the approach. Namely, it was demonstrated that the sum combination method did not achieve any results which were significantly better than that of standard GP. However, by using the vote method, this approach did manage to achieve significantly better results only for larger ensemble sizes. This approach also introduces an additional parameter into the GP approach, namely the bag size. In the experiments it was shown that the quality of the achieved results heavily depends on the value of this parameter, thus demonstrating the need to find an optimal value for it. Since the execution time of the procedure also depends on the bag size, the entire approach can be sped up by using smaller bag sizes. When using the bag size of 60 instances the execution speed is comparable to the execution speed of the BoostGP approach. Another benefit of this approach is that the evolved DRs which form the ensemble are completely independent. This makes it possible to run this approach in parallel on several computers and thus speeding up the entire process (for example if an ensemble of 10 DRs is evolved, the approach can be run once to evolve all ten DRs, or ten instances can be run in parallel where each instance evolves one DR).

Although the experiments were performed with bag sizes from those smaller than the standard problem set to those which were larger, the experiments did not give a conclusive answer as to which bag sizes were preferable by the procedure. However, on the average better results were achieved when using larger bag sizes.

Through the experiments one disadvantage for the BagGP approach was discovered. Namely the approach usually achieved the biggest dispersion among the solutions it found. This behaviour could prove to be problematic since this means that the approach will achieve solutions of variable quality.

7.3 BoostGP

Since the BoostGP approach offers additional information for each DR it evolves, namely its confidence, it was tested whether the inclusion of this information in the sum and vote combination methods improves the performance of this approach. Through the experiments it was shown that in almost all cases better results could be obtained when including the information about the confidences obtained by the BoostGP approach (especially when additionally using ESS). For both the weighted and unweighted approach it was shown that the sum method was unable to achieve significantly better results than the standard GP, while with the vote combination method this was possible for all tested ensembles of medium and larger sizes.

Although this approach achieves good results, it has several disadvantages. First of all this is the most complex approach when compared to the other approaches which were used. Secondly, unlike with the BagGP approach, the DRs which form one ensemble cannot be evolved in parallel. Although there is no direct dependency between the different DRs which are evolved, when one DR is evolved it will influence the weights which are used to determine the importance of the training instances when learning the new DR. Therefore, in this approach the rules need to be evolved sequentially one after another.

7.4 Cooperative coevolution

Although it was expected that the cooperative coevolution procedure would achieve good results, considering that it evolves DRs in dependence to other DRs that form the ensemble, the approach has achieved quite disappointing results for all criteria except the Nwt criterion. One possible explanation for such a behaviour could be that the procedure overfitted on the training set. This assumption is backed up by the fact that the cooperative coevolution approach achieves better results on the test set then the standard GP. Even trying out different termination criteria did not manage to improve the results significantly, thus in the future some other methods of preventing overfitting should be tried out in order to determine if this could improve the performance of the approach. Additionally, the few good results obtained by this approach were achieved mostly when using smaller ensemble sizes, which also suggests that the procedure struggles in evolving good DRs which complement the other DRs in the ensemble. This was especially evident for the vote combination method, where in certain situations the ensemble consisted of several rules which together made suboptimal choices. However, when one of those rules would be replaced, the effectiveness of the ensemble would deteriorate even further, therefore the algorithm would be stuck in a local optimum. Because of that reason, the procedure should be extended with mechanisms that could prevent such occurrences or correct them (for example by reinitialising the ensemble with random DRs).

The cooperative coevolution copes with another important problem, and that is its execution time. Namely, the execution time of this procedure heavily depends on the number of ensembles it evolves, but to a much greater extent than any of the aforementioned procedures. This is a consequence of the fact that in each iteration the cooperative coevolution approach has to evaluate an ensemble of DRs, thus prolonging the evaluation process, whereas the other procedures only evaluated individuals by themselves. This results in slower execution times especially for larger ensemble sizes.

Since the cooperative coevolution approach wasn't able to evolve bigger ensembles of good quality, the best subsets found by ESS were usually not better than the best solution found by the cooperative coevolution approach. Even by using ESS it was not possible to achieve results which would be significantly better than those of standard GP.

7.5 ESS

ESS has shown to be very promising in improving the results of the ensemble learning approaches. Naturally, ESS was not able to find a subset with a better quality than the original ensemble in every single occasion, but in many cases it was able to determine an ensemble subset which significantly improved the results when compared to the original ensemble. ESS has especially proven useful when being used with ensemble learning approaches which independently evolve the DRs that form the ensemble (BagGP and BoostGP). For SEC it was not able to significantly improve the results, since that approach is already similar to ESS. For cooperative coevolution ESS also did not achieve any improvements, however this could be due to the fact that in this approach DRs are much more interdependent than in other approaches (since the DRs are all evolved simultaneously). The best minimum values for all criteria, except the Cmax criterion, have been achieved by using ESS.

There are several benefits of using ESS. First of all it tries not only to find a better ensemble, but also to find an ensemble of a smaller size. As the experiments have shown, ESS was in many occasions able to find a better subset which significantly reduced the size of the original ensemble used by ESS. Secondly, it was shown that this approach is applicable to any of the tried out ensemble learning approaches, and that for some approaches (BagGP and BoostGP) it will be able to additionally improve their performances. Lastly, the execution time of this approach is fast even when performing an exhaustive search for ensembles of size ten. Naturally, with bigger ensembles the execution time of ESS would grow drastically, however it is questionable if there is even a need to evolve larger ensembles than the ones considered in this paper. Nevertheless, even if there would be a need to evolve larger ensembles, the execution time of ESS can be improved by not using an extensive search for the subsets of ensembles, but rather a random search or a search guided by some heuristic method.

Based on all the previous outlined characteristics, it is safe to conclude that ESS represents a good addition to ensemble learning approaches in order to improve its results and decrease the ensemble size.

7.6 Comparison of all approaches

This section gives a discussion on the complete results achieved by all the approaches. The cooperative coevolution approach has achieved the overall worst results, while the other approaches achieved good results depending on the considered criterion and algorithm parameters. The proposed SEC approach has demonstrated to be very effective and has achieved the best results for the Twt criterion. On the other criteria it also achieved results which were significantly better than those of standard GP. BagGP and BoostGP have shown to produce more dispersed results which in the end lead to the situation that for certain parameter combinations the two approaches are unable to find significantly better results than the standard GP. The cause for such dispersed results comes from the fact that the ensemble is formed by independently evolved and selected DRs. In order to try and increase the performance of the evolved ensembles, the ESS method was used. This method has shown to be able to improve the results of the BagGP and BoostGP approaches, while for the other two approaches it did not have a significant effect.

By comparing the vote and sum combination methods which were used by the ensemble learning approaches, it is hard to determine which of those two methods would be better. For the Twt and Nwt criteria the best overall results were achieved when using the sum combination method, for the Cmax criterion the best overall result was achieved when using the vote combination method, while for the Ft criterion both methods achieved the same best result. On the other hand, it was shown that in most cases when the ensemble learning approaches were using the vote combination method, they were able to achieve less dispersed results, than when using the sum combination method.

Regarding the ensemble sizes, it is hard to determine which of the ensemble sizes produces the best results, as this heavily depends both on the approach and the optimised criterion. If the cooperative coevolution approach is excluded (because of the problems it has with larger ensembles), most approaches usually achieve the best results when using ensembles of medium and large sizes. Therefore it seems to be advisable to use these approaches with such ensemble sizes. In addition to that, it was shown that the vote combination method performed better when using ensembles of larger sizes, while the sum combination method preferred smaller ensemble sizes.

8 Conclusion

This paper analysed the application of different ensemble learning approaches for creating ensembles of DRs: SEC, BagGP, BoostGP and cooperative coevolution. Through the carried out experiments it was shown that, by using the aforementioned ensemble learning approaches, it is possible to create ensembles of DRs which can significantly outperform the result obtained by the standard GP method. The best results were usually achieved by the SEC, BagGP and BoostGP approaches (depending on the criterion and ensemble size), while the worst results were clearly achieved by the cooperative coevolution approach. The proposed SEC approach has shown to be even more efficient than the BagGP and BoostGP methods, achieving not only better results, but also being able to create the ensembles much faster if previously generated DRs are available.

Furthermore, it was shown that it is possible to improve the results even further by using ESS to find the optimal subset of DRs to form the ensemble. The benefit of using ESS is not only in achieving better results, but also in reducing the ensemble size, which can improve interpretability, and speed up the dispatching process since a smaller number of DRs needs to be calculated. With the application of ESS it was possible to significantly improve results of BoostGP and BagGP, thus demonstrating the effectiveness of this approach. Therefore this approach represents a viable addition to the existing ensemble learning approaches.

In future studies the application of ensemble learning for creating ensembles of DRs will be studied further. One possible direction is to investigate what other ensemble learning approaches could be adapted and used for solving this problem. It would also be interesting to see if some other ensemble combination methods could be used, and how their results could compare to the results of the sum and vote combination methods. For the SEC approach it would be interesting to design heuristics which could be used for finding the optimal combination of ensembles (and maybe even the optimal size), rather than by using the random search used in this paper. Finally, it would also be interesting to try out the described procedures in the job-shop and other machine environments.

References

1. A. Allahverdi, J.N.D. Gupta, T. Aldowaisan, A review of scheduling research involving setup considerations. Omega **27**(2), 219–239 (1999). doi:10.1016/S0305-0483(98)00042-5
2. A. Allahverdi, C.T. Ng, T.C.E. Cheng, M.Y. Kovalyov, A survey of scheduling problems with setup times or costs. Eur. J. Oper. Res. **187**(3), 985–1032 (2008). doi:10.1016/j.ejor.2006.06.060
3. U. Bhowan, M. Johnston, M. Zhang, X. Yao, Reusing genetic programming for ensemble selection in classification of unbalanced data. IEEE Trans. Evolut. Comput. **18**, 893–908 (2013)
4. U. Bhowan, M. Johnston, M. Zhang, X. Yao, Evolving diverse ensembles using genetic programming for classification with unbalanced data. IEEE Trans. Evolut. Comput. **17**(3), 368–386 (2013)
5. J. Branke, S. Nguyen, C. Pickardt, M. Zhang, Automated design of production scheduling heuristics: a review. IEEE Trans. Evolut. Comput. (2015). doi:10.1109/TEVC.2015.2429314
6. L. Breiman, Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996). doi:10.1007/BF00058655
7. E.K. Burke, M.R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, J.R. Woodward, Exploring hyper-heuristic methodologies with genetic programming. Comput. Intell. **1**, 177–201 (2009). doi:10.1007/978-3-642-01799-5_6
8. E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, J.R. Woodward, A classification of hyper-heuristics approaches. Handb. Metaheuristics **57**, 449–468 (2010). doi:10.1007/978-1-4419-1665-5_15
9. C. Dimopoulos, A.M.S. Zalzala, A genetic programming heuristic for the one-machine total tardiness problem, in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, vol. 3, no. 1 (1999), pp. 2207–2214. doi:10.1109/CEC.1999.785549

10. M. Đurasević, D. Jakobović, K. Knežević, Adaptive scheduling on unrelated machines with genetic programming. Appl. Soft Comput. **48**, 419–430 (2016). doi:10.1016/j.asoc.2016.07.025
11. C. Ferreira, Gene expression programming : a new adaptive algorithm for solving problems. Complex Syst. **13**(2), 1–22 (2001)
12. G. Folino, C. Pizzuti, G. Spezzano, *GP Ensemble for Distributed Intrusion Detection Systems* (Springer, Berlin, 2005), pp. 54–62
13. G. Folino, C. Pizzuti, G. Spezzano, Training distributed GP ensemble with a selective algorithm based on clustering and pruning for pattern classification. IEEE Trans. Evolut. Comput. **12**(4), 458–468 (2008)
14. Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in *European Conference on Computational Learning Theory* (Springer, 1995), pp. 23–37
15. E. Hart, K. Sim, A hyper-heuristic ensemble method for static job-shop scheduling. Evolut. Comput. **24**(4), 609–635 (2016)
16. T. Hildebrandt, J. Heger, B. Scholz-Reiter, Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach, in *GECCO '10: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (2010), pp. 257–264. doi:10.1145/1830483.1830530
17. L. Hong, S.E. Page, Groups of diverse problem solvers can outperform groups of high-ability problem solvers. Proc. Natl. Acad. Sci. USA **101**(46), 16385–16389 (2004)
18. R. Hunt, M. Johnston, R. Hunt, M. Johnston, *Evolving "Less-myopic " Scheduling Rules for Dynamic Job Shop Scheduling with Genetic Programming*, pp. 927–934
19. H. Iba, Bagging, boosting, and bloating in genetic programming, in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation—Volume 2, GECCO'99* (Morgan Kaufmann Publishers Inc., 1999), pp. 1053–1060
20. D. Jakobović, Evolutionary computation framework. http://gp.zemris.fer.hr/ecf
21. D. Jakobović, Project site. http://gp.zemris.fer.hr/scheduling/
22. D. Jakobović, L. Jelenković, L. Budin, Genetic programming heuristics for multiple machine scheduling, in *Proceedings of the 10th European Conference on Genetic Programming*, vol. 4445 (2007), pp. 321–330. doi:10.1007/978-3-540-71605-1_30
23. D. Jakobović, K. Marasović, Evolving priority scheduling heuristics with genetic programming. Appl. Soft Comput. J. **12**(9), 2781–2789 (2012). doi:10.1016/j.asoc.2012.03.065
24. J.R. Koza, Human-competitive results produced by genetic programming. Genet. Program. Evolvable Mach. **11**(3–4), 251–284 (2010). doi:10.1007/s10710-010-9112-3
25. K. Miyashita, Job-shop scheduling with genetic programming, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)* (2000), pp. 505–512
26. S. Nguyen, K.C. Tan, *A Dispatching Rule Based Genetic Algorithm for Order Acceptance and Scheduling* (2015), pp. 433–440
27. S. Nguyen, M. Zhang, M. Johnston, *A Sequential Genetic Programming Method to Learn Forward Construction Heuristics for Order Acceptance and Scheduling* (2014), pp. 1824–1831. doi:10.1109/CEC.2014.6900347
28. S. Nguyen, M. Zhang, M. Johnston, K.C. Tan, A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems, in *2012 IEEE Congress on Evolutionary Computation, CEC 2012 (i)* (2012), pp. 10–15. doi:10.1109/CEC.2012.6252968
29. S. Nguyen, M. Zhang, K.C. Tan, Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems, in *2015 IEEE Congress on Evolutionary Computation (CEC)* (2015), pp. 2781–2788. doi:10.1109/CEC.2015.7257234
30. S. Nguyen, M. Zhang, M. Johnston, K.C. Tan, A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. IEEE Trans. Evolut. Comput. **17**(5), 621–639 (2013). doi:10.1109/TEVC.2012.2227326
31. S. Nguyen, M. Zhang, M. Johnston, K.C. Tan, Dynamic multi-objective job shop scheduling: a genetic programming approach, in *Automated Scheduling and Planning, Studies in Computational Intelligence*, vol. 505, ed. by A.S. Uyar, E. Ozcan, N. Urquhart (Springer, Berlin, 2013), pp. 251–282
32. S. Nguyen, M. Zhang, M. Johnston, K.C. Tan, Learning iterative dispatching rules for job shop scheduling with genetic programming. Int. J. Adv. Manuf. Technol. **67**(1–4), 85–100 (2013). doi:10.1007/s00170-013-4756-9

33. S. Nguyen, M. Zhang, M. Johnston, K.C. Tan, Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Trans. Evolut. Comput. **18**(2), 193–208 (2014). doi:10.1109/TEVC.2013.2248159
34. L. Nie, L. Gao, P. Li, L. Zhang, 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)* (2011), pp. 291–295. doi:10.1109/CSCWD.2011.5960088
35. L. Nie, X. Shao, L. Gao, W. Li, Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. Int. J. Adv. Manuf. Technol. **50**(5–8), 729–747 (2010). doi:10.1007/s00170-010-2518-5
36. G. Paris, D. Robilliard, C. Fonlupt, Applying boosting techniques to genetic programming, in *Artificial Evolution 5th International Conference, Evolution Artificielle, EA 2001*, vol. 2310 (2001), pp. 267–278. doi:10.1007/3-540-46033-0_22
37. J. Park, S. Nguyen, M. Zhang, M. Johnston, Genetic programming for order acceptance and scheduling, in *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, vol. 3 (2013), pp. 1005–1012. doi:10.1109/CEC.2013.6557677
38. J. Park, S. Nguyen, M. Zhang, M. Johnston, Evolving ensembles of dispatching rules using genetic programming for job shop scheduling. EuroGP **1**, 92–104 (2015)
39. F. Peng, K. Tang, G. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization. IEEE Trans. Evolut. Comput. **14**(5), 782–800 (2010)
40. M. Pinedo, *Scheduling Theory, Algorithms and Systems*, 4th edn. (Springer US, Boston, 2012)
41. R. Poli, W.B. Langdon, N.F. McPhee, A field guide to genetic programming. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk` (2008). (With contributions by J. R. Koza)
42. R. Polikar, Ensemble Learn. **4**(1), 2776 (2009)
43. M.A. Potter, K.A.D. Jong, A cooperative coevolutionary approach to function optimization, in *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, PPSN III* (Springer, London, 1994), pp. 249–257. http://dl.acm.org/citation.cfm?id=645822.670374
44. L.V.D. Souza, A.T.R. Pozo, A.C. Neto, J.M.C. Rosa, Genetic Programming and Boosting Technique to Improve Time Series Forecasting, in *Evolutionary Computation* (2009). doi:10.5772/9617
45. J.C. Tay, N.B. Ho, Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput. Ind. Eng. **54**(3), 453–473 (2008). doi:10.1016/j.cie.2007.08.008
46. S.Y. Yuen, X. Zhang, On composing an algorithm portfolio. Memet. Comput. **7**(3), 203–214 (2015)