Constructing ensembles of dispatching rules for multi-objective problems

Marko Đurasević[1], Lucija Planinić[1], Francisco J. Gil-Gala[2], and Domagoj Jakobović[1]

[1] Faculty of Electrical Engineering and Computing, University of Zagreb
[2] Department of Computer Science, University of Oviedo

Abstract. Scheduling represents an important aspect of many real-world processes, which is why such problems have been well studied in the literature. Such problems are often dynamic and require that multiple criteria be optimised simultaneously. Dispatching rules (DRs) are the method of choice for solving dynamic problems. However, existing DRs are usually implemented for the optimisation of only a single criterion. Since manual design of DRs is difficult, genetic programming (GP) has been used to automatically design new DRs for single and multiple objectives. However, the performance of a single rule is limited, and it may not work well in all situations. Therefore, ensembles have been used to create rule sets that outperform single DRs. The goal of this study is to adapt ensemble learning methods to create ensembles that optimise multiple criteria simultaneously. The method creates ensembles of DRs with multiple objectives previously evolved by GP to improve their performance. The results show that ensembles are suitable for the considered multi-objective problem.

Keywords: Genetic Programming, Scheduling, Unrelated Machines, Dispatching Rules, Ensembles, Multi-Objective optimisation

1 Introduction

Scheduling is the problem of optimally assigning a set of jobs to a finite number of machines [13]. Such problems have numerous applications in the real world, including manufacturing, universities, airports, hospitals, electric vehicle charging, and the like [13, 8]. Since most real-world scheduling problems are NP-hard, they are usually solved using various heuristic methods, most notably metaheuristics. However, metaheuristics are difficult to apply in dynamic environments. Therefore, simple heuristic methods called dispatching rules (DRs) are the method of choice for such problems. DRs construct the schedule online while the system is running by selecting which job to schedule next when a machine becomes available [17]. However, manually constructing such heuristics has proven to be a difficult and time-consuming task, leading to the use of various methods to automatically construct DRs [1].

Genetic programming (GP) is one of the most popular hyperheuristic methods used to develop new heuristics for various combinatorial problems. As such, it is widely used for automatic design of various scheduling problems. In most cases, using such a method, it has been possible to design new rules that outperform various manually developed rules. This led to a large amount of research considering different problem variants and methods for designing better DRs [1].

A key direction in automatic design of DRs is to develop rules that are also suitable for optimising multiple objectives, since most real-world problems usually require optimising multiple objectives [10]. This direction is important because manually developed rules are usually only suitable for optimising a single criterion. This, of course, requires the application of different multi-objective algorithms (MO) to develop rules for different combinations of criteria under consideration. Although previous studies have shown that efficient MO DRs can be designed using GP [16], these rules are limited by the aspect that a single rule still performs poorly in certain situations. One of the most efficient ways to improve the performance of DRs is to combine them into ensembles of rules that make decisions together.

In this study, we are concerned with the application of ensemble learning methods to create ensembles of DRs suitable for simultaneous optimisation of multiple criteria. We are interested in answering the question of whether it is possible to improve the performance of DRs designed for optimising multiple objectives by combining them into ensembles. Therefore, an ensemble learning method is adapted for the MO case and used to create ensembles for a MO problem. The contributions of this study can be summarised as follows:

1. Adapt an ensemble learning method to construct ensembles for MO problems
2. Analyse the performance of the proposed method on a selected MO problem
3. Examine the Pareto fronts obtained by the different MO methods

The rest of the paper is organised as follows. Section 2 provides the overview on the existing literature. The unrelated machines problem and automatic design of dispatching rules for it are described in Section 3. The ensemble learning method adapted for MO problems is outlined in Section 4. The obtained results are shown in Section 5. Finally, the conclusion and future research directions.

2 Literature review

A MO problem for the job shop environment, where five criteria must be optimised simultaneously, was considered in [9], where several MO algorithms were used. This research was extended in [10], where the authors applied a local search during evolution. In [7], several MO algorithms were used to develop rules for problems involving four and five criteria. In [16], the authors consider several MO problems with 3 to 9 criteria and apply 4 MO GP algorithms to evolve new DRs for them. In [20], the authors consider the dynamic flexible job shop problem and apply the NSGA-II and SPEA2 algorithms to construct DRs for multiple

flowtime objective formulations. In [6], the authors analyse how different features of the job-shop problem affect the performance of a MO GP method.

In [12], the authors apply the cooperative coevolution algorithm to create ensembles for the job shop environment. In this approach, each subpopulation in the algorithm represents a rule that is evolved for the ensemble. In [5], a method called NELLI-GP was proposed for evolving ensembles of DRs, which achieves better results than the cooperative coevolution from [12]. In [11], four ensemble combination methods were investigated: sum, weighted sum, voting, and weighted voting. Four ensemble learning methods were compared in [15], including BagCP, BoostGP, cooperative coevolution, and SEC. Since the SEC method performed the best, the study was extended in [18], where the SEC method was analysed in more detail. In [14], the ensemble learning methods from [15] were applied to the resource constrained project scheduling problem. Another type of ensembles was proposed in [4, 3] for the variable capacity one machine problem. These ensembles use each rule to construct the schedule individually and then select the best results, making them suitable for static environments.

3 Background

3.1 Unrelated machines environment

The unrelated machines environment is a scheduling problem consisting of n jobs that have to be scheduled on a given set of m machines. Each job i is defined by its processing time p_{ij} on machine i, its ready time r_j, its due date d_j, and its weight w_j. The problem is considered under dynamic conditions, i.e., it is not known a priori when the jobs will be fed into the system, nor are job characteristics known before the jobs' ready time. When each job is scheduled, the completion time C_j and the tardiness of a job T_j can be calculated. The tardiness represents how much job j spent executing after its due date, and is defines as $T_j = \max(C_j - d_j, 0)$. Based on the previous properties, several scheduling criteria can be defined which will be considered in this study:

- C_{max} - maximum completion time of all jobs: $C_{max} = \max_j\{C_j\}$.
- Cw - total weighted completion time: $Cw = \sum_j w_j C_j$,
- Twt - total weighted tardiness: $Twt = \sum_j w_j T_j$,

All of the above criteria need to be minimised simultaneously. Therefore, the problem considered in this paper can be defined as $R|r_j|C_{max}, Cw, Twt$ using the standard notation for scheduling problems [13].

3.2 Automatic design of DRs with GP

DRs are constructive heuristics consisting of a schedule generation scheme (SGS) and a priority function (PF). The SGS is a procedure that builds the schedule so that each time a machine becomes available, it selects which job to schedule next. However, the decision of which job to schedule on which machine is not

made by the SGS itself; instead, the PF is used to make this decision. The PF is used to assign a numeric value to each job-machine pair, and the pair that received the smallest value is selected for scheduling.

Traditionally, PFs were designed manually, resulting in a plethora of DRs to optimise various criteria. However, another option is to design such PFs using GP. Since PF is a mathematical expression, it can be easily coded as a solution tree in GP. For this purpose, however, a set of terminal and function nodes must be defined that GP can use to construct the expression. The terminal nodes used are listed in Table 1. These nodes represent important values of the system like job processing times, due dates, average execution times, and the like. For the function set, the addition, subtraction, multiplication, protected division (returns 1 if division is by 0), and unary positive operators ($pos(a) = \max(a, 0)$). These were selected based on a previous study [19].

Table 1: Terminal set

Terminal	**Description**
pt	processing time of job j on machine i
$pmin$	minimal processing time (MPT) of job j
$pavg$	average processing time of job j across all machines
PAT	time until machine with the MPT for job j becomes available
MR	time until machine i becomes available
age	time which job j spent in the system
dd	time until which job j has to finish with its execution
w	weight of job j (w_j)
SL	slack of job j, $-max(d_j - p_{ij} - t, 0)$

4 Designing ensembles for MO problems

To design ensembles of DRs, two things must be specified: how the ensemble is constructed and how the DRs that make up the ensemble work together.

4.1 Ensemble construction

Although a variety of ensemble construction methods have been proposed for constructing ensembles of DRs, the simple ensemble combination method (SEC) was chosen. The reason for choosing this method is its inherent simplicity and better results compared to other ensemble learning methods. The idea of the method is that it randomly constructs a set of ensembles from a pool of existing DRs and returns the best ensemble as the result.

Algorithm 1 shows the SEC method adapted for MO optimisation. The method accepts three parameters, the number of ensembles N to construct, the

number of rules in ensemble ES, and the set of rules used to construct ensembles R. In each iteration, an ensemble is constructed by randomly selecting ES rules from the set R. When the ensemble is constructed, it is added to the set *constructed* that contains all constructed ensembles. This procedure is repeated N times, resulting in a set containing N ensembles. This set is then nondominately sorted [2] for the specified criteria, and the first front is returned by the SEC method. In this way, a set of solutions that provide different tradeoffs between the considered criteria should be obtained.

Algorithm 1 The simple ensemble combination method

```
1: function SEC(N, ES, R)
2:     constructed ← ∅
3:     counter ← 0
4:     while counter < N do
5:         counter + +
6:         E ← ∅
7:         while |E| < ES do
8:             Select a random DR from R \ E, and add it to E
9:         end while
10:        constructed ← ensembles ∪ {E}
11:    end while
12:    Perform nondominated sorting on the constructed set
13:    return first front of constructed
14: end function
```

4.2 Ensemble combination

When a set of rules is chosen to form an ensemble, it is important to specify how these rules will work together, i.e., how their individual decisions will be combined into a single decision. For this purpose, sum and vote combination methods are used [15]. At each decision point, these methods combine the decisions of all the individual rules in the ensembles into a single decision, which is then executed by the SGS. Since each rule computes a numerical priority value for job-machine pairs, the easiest way to aggregate the decisions of the individual DRs is to add the priority values obtained from each rule. Then the job-machine pair with the lowest value is selected and scheduled. This is how the sum combination method works. An obvious pitfall with this method is that the rules in the ensemble can produce very different priority values, which could result in one rule dominating over others. Therefore, the vote combination method uses a simple voting mechanism in which each rule in the ensemble casts a vote for the job-machine pair to be scheduled (the one with the lowest priority value), and the one that received the most votes is eventually scheduled. One problem with this approach is that ties can occur, and while they can be resolved in different ways, they are resolved in such a way that the job that arrived first is selected.

5 Experimental analysis

5.1 Setup

To test the effectiveness of the adapted SEC method, we evaluate it on the $R|r_j|C_{max}, Cw, Twt$ problem. However, to apply the SEC method, a pool of MO DRs that can be combined into ensembles is required. To obtain this pool of rules, first the considered problem is optimised using the NSGA-II, NSGA-III, HaD-MOEA, and MOEA/D and then all obtained solutions are combined into a single set of nondominated solutions [16]. After this first step, 70 MO DRs were obtained, which are then used by SEC to construct the ensembles.

The SEC method is tested with different parameter values. The values 1000, 5000, and 10000 are used for the number of iterations of the method. Both the sum and vote combination methods are tested with ensemble sizes of 3, 5, and 7. For each parameter combination, the method is executed 30 times and the best Pareto front of each execution is saved. To evaluate the quality of the obtained Pareto fronts, the hypervolume (HV) metric is used [16]. The reason for choosing this metric is that it measures both convergence and diversity of Pareto fronts.

To create and evaluate the ensembles, a set of 120 instances was used [19]. The set was split into two different sets, the training set and test set. The training set was used to develop rules and create ensembles. The test set was then used to evaluate the performance of the constructed rules and ensembles.

5.2 Optimising the $R|r_j|C_{max}, Cw, Twt$ problem

Table 2 shows the HV values obtained for all tested parameter combinations of the studied SEC method. The row labelled "med" represents the median HV value obtained for each algorithm execution. The line "tot", on the other hand, denotes the HV value of the union of all 30 Pareto fronts obtained in the executions. The last lines denotes the HV value obtained for the set of individual MO rules used by SEC. The best values for each method are in bold. The results illustrate several things. First, the vote combination method clearly achieves better results than the sum method. As for the size of the ensemble, it is difficult to determine which size would be the best. The sum method performs best when 7 is used, but with the vote method, there is no single ensemble size that gives the best results. Second, one can see that the SEC method in a single execution gets Pareto fronts with lower HV values than the front of the MO DRs. However, if we consider the union of these fronts obtained in each execution, SEC always obtained a better Pareto front, at least for the vote method. This seems to indicate that a single execution of SEC is not sufficient to obtain a good approximation of the Pareto front, but rather that multiple executions are required.

To get a better idea of the differences between the Pareto fronts of the MO DRs and the ensembles, the solutions in their respective Pareto fronts are shown in Figure 1. The figure shows pairwise combinations of the three optimised criteria to better illustrate the Pareto fronts. For the ensemble, the Pareto front

Table 2: Results for the HV metric for the $R|r_j|C_{max}, Cw, Twt$ problem

Method		sum 3	sum 5	sum 7	vote 3	vote 5	vote 7
SEC-1000	med	0.568	0.565	0.563	**0.588**	0.587	0.586
	tot	0.591	0.593	0.603	**0.649**	0.634	0.622
SEC-5000	med	0.572	0.566	0.567	0.593	**0.596**	0.591
	tot	0.590	0.593	0.597	0.632	0.647	**0.651**
SEC-20000	med	0.571	0.571	0.571	0.598	**0.599**	0.596
	tot	0.589	0.593	0.599	0.622	**0.640**	0.622
MO DRs				0.604			

of SEC-5000 was used with the vote combination method and an ensemble size of 7 rules, since the best Pareto front was obtained for this parameter combination. For all three combinations, the figure shows that the Pareto front of the ensembles is closer to the origin of the coordinate system, indicating that much better convergence was achieved. However, one problem with this Pareto front could be that more solutions appear to be grouped together, suggesting that the algorithm favours more convergence than diversity.

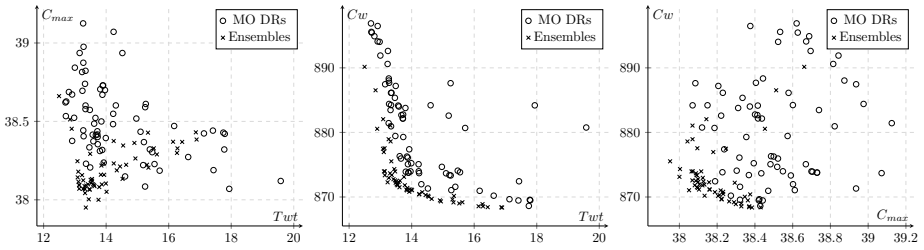


Fig. 1: The Pareto front obtained for the $R|r_j|C_{max}, Cw, Twt$ problem denoted through pairwise combinations of the three optimised criteria

Table 3 shows the descriptive statistics calculated for the obtained Pareto fronts for SEC-5000. The table shows the dominance of the vote combination method, as it obtains better median and minimum values than the sum method for all criteria. Compared to the MO DRs given in the end of the table, the Pareto fronts of the ensembles generally achieve better median and maximum values. For the minimum values, only the sum method achieved better minimum values than the individual MO DRs.

Figure 2 shows the box plots of the optimised criteria for the Pareto fronts obtained by different methods. The results obtained by ensembles are denoted by E-x-y, where "x" stands for "s" or "v" depending on whether the sum or vote

Table 3: Descriptive statistics of the Pareto fronts obtained for the $R|r_j|C_{max}, Cw, Twt$ problem

Method		sum			vote		
		min	med	max	min	med	max
E-3	C_{max}	38.13	38.39	39.00	37.95	**38.31**	38.68
	Cw	868.8	872.6	894.8	867.9	**871.8**	894.0
	Twt	13.04	14.16	17.67	12.78	**13.93**	17.04
E-5	C_{max}	38.19	38.33	38.97	37.92	**38.25**	39.11
	Cw	869.2	873.1	885.5	868.2	**870.8**	893.6
	Twt	12.96	**13.85**	16.27	12.58	14.36	17.15
E-7	C_{max}	38.18	38.36	38.88	37.95	**38.16**	38.66
	Cw	868.9	872.2	885.1	868.4	**871.8**	890.2
	Twt	12.94	13.99	16.75	12.49	**13.63**	16.89
MO DRs	C_{max}	38.07	38.46	39.13	38.07	38.46	39.13
	Cw	868.6	880.7	896.9	868.6	880.7	896.9
	Twt	12.70	13.83	19.58	12.70	13.83	19.58

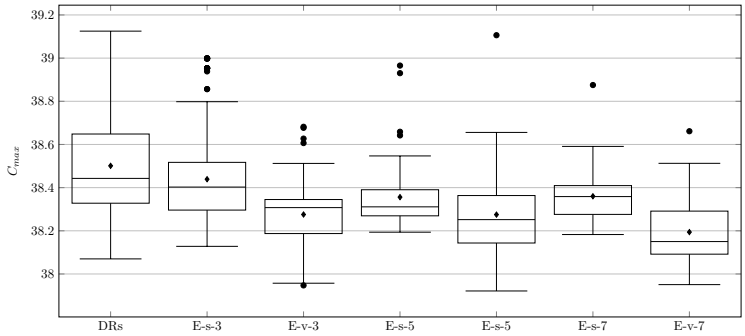
combination method is used, and "y" stands for the size of the ensemble. For the C_{max} criterion, we can see that all ensemble variants achieve a better distribution of solutions. This is especially true for the vote combination method. Something similar can be observed for the Cw criterion, with even larger differences. However, for the Twt criterion, in several cases the ensembles obtained Pareto fronts with worse distributions than the MO DRs. In the other cases, the distribution of solutions is mostly similar to the MO DRs, or the values obtained for the criterion are slightly less scattered.

6 Conclusion

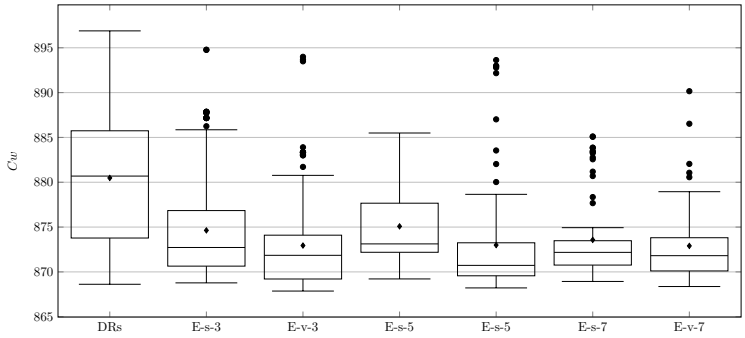
This study addresses the application of ensemble learning to create ensembles of MO DRs to optimise multiple criteria simultaneously. The SEC method, previously used only for the single objective case, was adapted for multi-objective problems and tested on the $R|r_j|C_{max}, Cw, Twt$. The obtained results show that the constructed ensembles can outperform the results obtained by single MO DRs. The analysis of the Pareto fronts showed that the constructed ensembles achieve much better convergence and still provide good coverage of the search space. In future studies, we plan to explore the possibility of using rules developed for a single target to construct MO ensembles and extend the experiments to more MO problems.

Acknowledgements

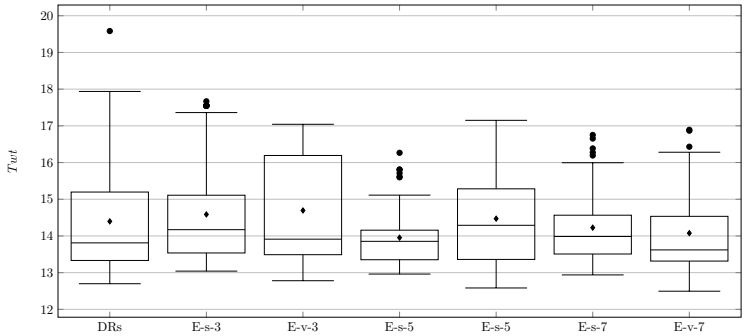
This research has been supported by the Spanish Government under research project PID2019-106263RB-I00, and by the Croatian Science Foundation under the project IP-2019-04-4333.



(a) Solution distributions for the C_{max} criterion



(b) Solution distributions for the Cw criterion



(c) Solution distributions for the Twt criterion

Fig. 2: Distribution of solutions for all considered criteria.

References

1. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated design of production scheduling heuristics: A review. IEEE Transactions on Evolutionary Computation 20(1), 110–124 (2016)
2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6(2),

182–197 (2002)
3. Gil-Gala, F.J., Mencía, C., Sierra, M.R., Varela, R.: Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms. Integrated Computer-Aided Engineering 28(1), 65–80 (Dec 2020), https://doi.org/10.3233/ICA-200634
4. Gil-Gala, F.J., Sierra, M.R., Mencía, C., Varela, R.: Combining hyper-heuristics to evolve ensembles of priority rules for on-line scheduling. Natural Computing (Jun 2020), https://doi.org/10.1007/s11047-020-09793-4
5. Hart, E., Sim, K.: A hyper-heuristic ensemble method for static job-shop scheduling. Evolutionary Computation 24(4), 609–635 (2016)
6. Masood, A., Chen, G., Mei, Y., Al-Sahaf, H., Zhang, M.: Genetic programming with pareto local search for many-objective job shop scheduling. In: AI 2019: Advances in Artificial Intelligence, pp. 536–548. Springer International Publishing (2019), https://doi.org/10.1007/978-3-030-35288-2_43
7. Masood, A., Mei, Y., Chen, G., Zhang, M.: Many-objective genetic programming for job-shop scheduling. pp. 209–216 (07 2016)
8. Mencía, C., Sierra, M.R., Mencía, R., Varela, R.: Evolutionary one-machine scheduling in the context of electric vehicles charging. Integrated Computer-Aided Engineering 26(1), 49–63 (Dec 2018), https://doi.org/10.3233/ica-180582
9. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Dynamic multi-objective job shop scheduling: A genetic programming approach. In: Studies in Computational Intelligence, pp. 251–282. Springer Berlin Heidelberg (2013)
10. Nguyen, S., Zhang, M., Tan, K.C.: Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. In: 2015 IEEE Congress on Evolutionary Computation (CEC). pp. 2781–2788 (2015)
11. Park, J., Mei, Y., Nguyen, S., Chen, G., Zhang, M.: An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. Applied Soft Computing 63 (11 2017)
12. Park, J., Nguyen, S., Zhang, M., Johnston, M.: Evolving ensembles of dispatching rules using genetic programming for job shop scheduling. pp. 92–104 (04 2015)
13. Pinedo, M.L.: Scheduling. Springer US (2012), https://doi.org/10.1007/978-1-4614-2361-4
14. Đumić, M., Jakobović, D.: Ensembles of priority rules for resource constrained project scheduling problem. Applied Soft Computing 110, 107606 (2021)
15. Đurasević, M., Jakobović, D.: Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. Genetic Programming and Evolvable Machines 19(1-2), 53–92 (Apr 2017)
16. Đurasević, M., Jakobović, D.: Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. Genetic Programming and Evolvable Machines 19(1-2), 9–51 (Sep 2017)
17. Đurasević, M., Jakobović, D.: A survey of dispatching rules for the dynamic unrelated machines environment. Expert Systems with Applications 113, 555–569 (2018)
18. Đurasević, M., Jakobović, D.: Creating dispatching rules by simple ensemble combination. Journal of Heuristics 25(6), 959–1013 (May 2019), https://doi.org/10.1007/s10732-019-09416-x
19. Đurasević, M., Jakobović, D., Knežević, K.: Adaptive scheduling on unrelated machines with genetic programming. Applied Soft Computing 48, 419–430 (2016)
20. Zhang, F., Mei, Y., Zhang, M.: Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 1366–1373 (2019)