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

Evolving Dispatching Rules for Optimising Many-Objective Criteria in the Unrelated Machines Environment

Marko Đurasević · Domagoj Jakobović

Received: date / Accepted: date

Abstract Dispatching rules are often a method of choice for solving various scheduling problems. Most often, they are designed by human experts in order to optimise a certain criterion. However, it is seldom the case that a schedule should optimise a single criterion all alone. More common is the case where several criteria need to be optimised at the same time. This paper deals with the problem of automatic design of dispatching rules by the use of genetic programming, for many-objective scheduling problems. Four multi-objective and many-objective algorithms, including nondominated sorting genetic algorithm II (NSGA-II), nondominated sorting genetic algorithm III (NSGA-III), harmonic distance based multi-objective evolutionary algorithm (HaD-MOEA) and multi-objective evolutionary algorithm based on decomposition (MOEA/D), have been used in order to obtain sets of Pareto optimal solutions for various many-objective scheduling problems. Through experiments it was shown that automatically generated multi-objective DRs not only achieve good performance when compared to standard DRs, but can also outperform automatically generated single objective DRs for most criteria combinations.

Keywords Dispatching rules · genetic programming · many-objective optimisation · scheduling · unrelated machines environment

1 Introduction

Scheduling can be defined as a decision-making process concerned with the allocation of scarce resources to tasks over a given time period, in order to optimise one or more objectives [64]. Scheduling problems appear in many real life domains, such as airplane scheduling in air traffic control [12], [23]; semiconductor manufacturing [63]; and therapy scheduling in hospitals [61]. Unfortunately, most scheduling problems belong to the category of NP-hard problems. This means that an algorithm which could find optimal solutions for such problems, in a reasonable amount of time, does

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

not exist. Because of that, scheduling problems are usually solved by using different heuristic algorithms. In this context, heuristic algorithms are mostly divided into two categories: the first group consists of metaheuristics which search the entire space of solutions (schedules) in order to find the best one, while the second group consists of problem specific heuristics (mostly in the form of dispatching rules) that incrementally construct the schedule.

Since most scheduling problems are combinatorial by nature, *search-based* metaheuristic methods (such as genetic algorithms, particle swarm optimisation, etc.) can be used to search the solution space [61], [24], [3], [13]. These methods have the advantage that they are able to find high quality solutions for many different scheduling problems. On the downside, they require a substantial amount of time in order to find solutions of acceptable quality. As a consequence, these approaches are mostly not applicable in dynamic scheduling problems, where there is a need for constant adaptation to the changing conditions of the scheduling environment (e.g. unplanned arrival of new jobs, machine breakdowns).

On the other hand, dispatching rules (DRs) do not search the entire space of solutions, but instead incrementally build the entire solution (schedule) [47], [7], [40]. The advantage of such heuristics is that they are able to construct schedules in time which is almost negligible when compared to the search-based approaches [68]. Because of their fast execution time and their ability to quickly react to changing conditions, DRs are applicable in dynamic scheduling environments. However, DRs also cope with a certain number of problems. First of all, the quality of schedules constructed by DRs is in most cases not as good as of those found by the search-based approaches. This is to be expected considered that these heuristics do not search the entire solution space. A second problem is that DRs are very hard to design, and that they are mostly designed for optimizing a few common objectives from the literature. Since it is complicated to design new DRs, they are mostly designed to optimise only one or two objectives, although there is often a need to optimise several objectives at the same time. As a consequence, DRs which optimise a user specified set of several criteria might not even exist, and would have to be designed by domain experts, which would be a lengthy and costly process. Lastly, since many DRs exist, it is not always obvious which of the existing DRs is the most appropriate for the given scheduling problem instance. This has been demonstrated by Branke and Pickardt in [6], where they use a genetic algorithm in order to generate job-shop problem instances, for which some standard DRs make sub-optimal decisions.

In order to deal with some of the aforementioned disadvantages, genetic programming (GP) [65], [44] is commonly used in order to automatically generate DRs. The DRs which are generated by GP are shown, in most cases, to be able to create schedules of better or similar quality, when compared to standard DRs. Additionally, by using GP for generating DRs, it is possible to create rules for arbitrary scheduling criteria relatively fast and for a wide variety of scheduling problems. An additional benefit of GP comes from the fact that there is a wide variety of evolutionary algorithms which deal with multi-objective and many-objective optimisation, and they have shown to achieve good performances on a large number of problems [74]. Therefore, by coupling GP with such procedures, it is possible to automatically design DRs which are able to simultaneously optimise several scheduling objectives. This demonstrates that there is a great amount of flexibility available when producing DRs by the use of GP.

Unfortunately, multi-objective GP algorithms were used only in few occasions to create DRs which optimise several scheduling criteria simultaneously. They were applied for optimising one multi-objective problem consisting of five scheduling criteria [53], [57], and recently also a problem consisting of four scheduling criteria in which one objective was removed from the original five [48]. Because of that, the aim of this paper is to examine the effectiveness of GP for creating DRs for many-objective scheduling problems consisting of different scheduling criteria. For that purpose several popular algorithms from the literature have been selected and used in order to create DRs for many-objective criteria. The algorithms will be applied on several criteria combinations consisting of three, five, seven and nine criteria. Although optimising seven objective problems is rarely found in the scheduling literature, there are a few instances in which six objective scheduling problems were considered [70] [11], and because of that several seven objective problems were also considered in this paper. The nine objective problem was included in order to analyse the correlation of all nine criteria at the same time. To analyse the performance of the automatically developed DRs, several DRs will be selected and compared to four standard DRs and also the best results achieved by a single objective GP. Finally, since the criteria set being optimised has shown to significantly influence the performances of many-objective algorithms, an analysis on how different criteria correlate with each other is also given.

The objectives of this paper are the following:

1. Automatic generation of multi-objective DRs and comparison with standard DRs and automatically generated single objective DRs.
2. Analysis of the mutual correlation of different scheduling criteria.
3. Comparison of MOGP algorithms for evolving DRs for the unrelated machines scheduling problem.

The first point will give the answer to the question whether automatically generated DRs for optimising several objectives simultaneously can outperform or perform equally well as manually designed DRs and automatically generated DRs for optimising a single objective. It will also be analysed if the criteria combinations affect whether automatically generated multi-objective DRs are able to perform better than single objective DRs. Through the second point we give an answer to the question how the different scheduling objectives influence each other when they are optimised together. Finally, the last point will give the answer to which multi-objective algorithms performed best for which criteria combinations, and why.

The remainder of the paper is organised as follows: Section 2 gives an overview of the current research in the field of automatic design of DRs. The unrelated scheduling environment is described in Section 3, while the genetic programming method for generating DRs is described in Section 4. Section 5 gives a description of the experimental design and the parameter optimisation process which was performed, while the achieved results and the discussion are given in section 6. Finally, section 7 gives a short conclusion and outlines some future work directions.

2 Literature Overview

Since its discovery GP has often been used in the field of hyper-heuristics, because it is able to represent and evolve complex expressions and rules [10], [9]. Consequently,

GP has also been extensively used to evolve new DRs for various scheduling problems. Dimopoulos and Zalzala used GP to evolve DRs for the one machine environment and showed that the evolved DRs achieved better results than some traditional rules [17]. Miyashita used GP to evolve rules for the job-shop scheduling environment [49]. He considered the scheduling problem as a multi-agent problem (where an agent denoted a machine), and based on that he proposed three different models. The first model was a homogeneous model which evolved the same DR for all agents. The distinct agent model, on the other hand, evolved DRs for each agent separately. The mixed agent model tried to combine the previous two models in a way that it develops two different DRs. Which of these two DRs would be assigned to a resource would depend on whether the machine is a bottleneck or not. Although the mixed agent model achieved the best results, it requires prior knowledge about the scheduling environment in order to detect the bottleneck machines. Jakobović *et al.*, apart from evolving dispatching rules for the single machine and job-shop scheduling environments, have also proposed a GP method (GP-3) which extends the mixed agent model of Miyashita [35]. Their approach generates three expressions, first of which represents the DR used for bottleneck machines, the second represents the DR used non bottleneck machines, and the third a decision function used to determine whether a machine represents a bottleneck or not. Therefore this approach does not need any prior knowledge about the scheduling environment, since it uses the decision function in order to adapt to the changing conditions during the execution of the system. Although the GP-3 method does not guarantee that the expression it evolved will truly be able to identify which machines are bottlenecks, it was nevertheless able to achieve better results when compared to the approach which evolved only one DR for all machines. Apart from the already mentioned machine environments, GP was also used to design DR for the parallel machines environment [36]. Hildebrandt *et al.* have performed an extensive analysis of creating DRs for the dynamic job-shop environment [27].

Gene expression programming (GEP) [20], a methodology similar to GP, was used by Nie *et al.* for evolving DRs in the single-machine [59] and job-shop [58] environments. Jakobović and Marasović further investigated the single-machine and job-shop environments, especially concerning the influence of algorithm parameters on the quality of the evolved DRs [37]. Additionally, in the same paper, the authors evolve DRs for the single-machine environment with set-up times and precedence constraints, and show that they achieve better results than several standard rules, thus demonstrating that GP is applicable for creating good DRs even for more complicated environments. Different solution representations of DRs have been analysed by Nguyen *et al.* in [52], showing that different GP representations can have a significant influence on the quality of DRs. Nguyen *et al.* also proposed a new type of GP for static scheduling problems, which constructs iterative dispatching rules (IDRs) and shows superior performance when compared to the simple GP approach [54]. The problem of global perspective of DRs has been analysed in [28]. In [29] it was shown that GP was able to evolve optimal dispatching rules for the static two machine job-shop environment, demonstrating the ability of GP to discover high quality solutions. Đurasević *et al.* have used several GP approaches (standard GP, GEP [20], IDRs, dimensionally aware GP [41]) in order to evolve DRs for the unrelated machines environment, and compare the effectiveness of the generated DRs [19]. GP has also been used for developing DRs for the order acceptance and scheduling (OAS) problem, where it was also able to evolve DRs better than some standard rules [60],

[50], [56]. This only shows that GP is not only applicable for evolving standard DRs, but also for similar scheduling problems.

Creating DRs for multi-objective and many-objective optimisation criteria has rarely been touched upon. Tay and Ho [67] were the first to use GP for evolving DRs for multi-objective criteria. In their case, the multi-objective problem was transformed into a single objective problem by linearly combining all three individual objectives. Therefore, standard single objective GP approaches could be used in order to evolve DRs for this problem. More recently, Nguyen *et al.* used several multi-objective GP approaches in order to evolve scheduling rules consisting of a DR and due-date assignment rule [51], [55]. Those approaches evolved two expression trees, one of which would be used for due-date assignment to jobs, while the other would be used as a standard dispatching rule. The results showed that the evolved rules outperformed various combinations of existing scheduling rules from the literature. Evolving dispatching rules for multi-objective criteria was also analysed by Nguyen *et al.* in [53] and [57]. In their studies, Nguyen *et al.* evolved DRs for optimizing five scheduling criteria simultaneously by using the HaD-MOEA algorithm. They showed that very efficient dispatching rules could be evolved even for such many-objective problems. GP has also shown to achieve good results for task assignment in assembly line balancing, where it achieved better results than many standard and metaheuristic approaches [4]. In [48] authors have also used several multi-objective algorithms in order to evolve DRs for four and five objectives.

A recent survey by Branke *et al.* gives a more detailed overview of all the applications of GP for evolving new dispatching rules [5].

Regarding multi-objective and many-objective optimisation in the unrelated machines environment, not much research has been done in this area. Fowler *et. al.* considered the problem of scheduling a printed wiring board manufacturer's drilling operation subject to five optimisation criteria, and they have used several approaches in order to solve the given problem [21]. In [70] the authors have optimised a scheduling problem of a printed wiring board manufacturing line, where six scheduling criteria were optimised simultaneously. The simulated annealing approach was used by Kolahan and Kayvanfar, in order to solve a scheduling problem consisting of the makespan, earliness and tardiness cost, and matching cost objectives [43]. In [46] a scheduling problem consisting of two and three scheduling criteria was optimised by the use of two proposed heuristics and a genetic algorithm. A short overview of some other multi-objective problems in the unrelated machines environment can be found in [62]. Although the research on this topic is quite sparse, the references show that multi-objective and many-objective optimisation problems in the unrelated machines environment appear in many real world situations, therefore outlining the need for development of DRs which are suited to optimise several criteria simultaneously.

3 The Unrelated Machines Environment

The unrelated machines environment consists of n jobs which compete in order to be scheduled on one of the m available machines [64]. Each job consists of several properties, including: processing time p_{ij}, which determines the processing time needed to execute the job with the index j on the machine with the index i; release time r_j, which determines when job j becomes available for scheduling; due date d_j, which determines the point in time until when the job with index j should finish with the

execution to not invoke any loss; and weight w_j which determines the weight (or importance) of the job with index j. In this paper different weights will be used for the tardiness (denoted as w_{Tj}), earliness (denoted as w_{Ej}) and completion times (denoted as w_{Cj}) criteria. The application of the unrelated machines environment can be found in different multiprocessor systems or production environments.

3.1 Scheduling Criteria

Many different evaluation criteria have been defined for scheduling problems in the literature. In this study, the eight most prominent criteria from the literature have been selected and were optimised in different combinations by the multi-objective algorithms.

When a job is scheduled, it is possible to calculate several metrics of that job, which will then in turn be used for defining metrics of the entire schedule:

- C_j - completion time of job j
- F_j - flowtime of job j:
$$F_j = C_j - r_j. \tag{1}$$
- T_j - tardiness of job j:
$$T_j = \max\{C_j - d_j, 0\}. \tag{2}$$
- E_j - earliness of job j:
$$E_j = \max\{-(C_j - d_j), 0\}. \tag{3}$$
- U_j - flag if a job is tardy or not:
$$U_j = \begin{cases} 1 : T_j > 0 \\ 0 : T_j = 0 \end{cases}. \tag{4}$$

By using the aforementioned job metrics, criteria for the complete schedule can be defined [1], [2]:

- C_{max} - maximum completion time of all jobs:
$$C_{max} = \max_j\{C_j\}. \tag{5}$$
- F_{max} - maximum flowtime:
$$F_{max} = \max_j\{F_j\}. \tag{6}$$
- T_{max} - maximum tardiness:
$$T_{max} = \max_j\{T_j\}. \tag{7}$$
- Cw - total weighted completion time:
$$Cw = \sum_j w_{Cj} C_j, \tag{8}$$

– Twt - total weighted tardiness:

$$Twt = \sum_j w_{Tj} T_j, \tag{9}$$

– Ft - total flowtime:

$$Ft = \sum_j F_j, \tag{10}$$

– Nwt - weighted number of tardy jobs:

$$Nwt = \sum_j w_{Tj} U_j. \tag{11}$$

– $Etwt$ - weighted earliness and weighted tardiness:

$$Etwt = \sum_j (w_{Ej} E_j + w_{Tj} T_j), \tag{12}$$

– M_{ut} - machine utilisation:

$$M_{ut} = \max_i \{\frac{P_i}{C_{max}}\} - \min_i \{\frac{P_i}{C_{max}}\}, \tag{13}$$

where P_i is defined as the sum of processing times of all jobs which were executed on machine with index i.

The last criterion, denoted as *machine utilisation*, is an additional criterion which we defined for this study. The idea behind this criterion is that by optimizing it the load is evenly distributed to all machines, in order to avoid the case in which some machines do little processing, while others are overloaded. Although machine utilisation is rarely used as the main scheduling criterion, in scenarios where load balancing is important it represents an essential secondary criterion.

3.2 Scheduling Conditions

Based on the availability of job parameters, scheduling can be performed in different conditions. In case that all parameters are known in advance, the schedule can be built before the system starts with its execution. This type of scheduling is called static scheduling. Search-based methods are most commonly used to create schedules for this type of scheduling.

On the other hand, if the information about the jobs is not available until they arrive into the system, then such scheduling is called dynamic scheduling. DRs are most commonly used in this kind of scheduling conditions, since they quickly react to the sudden changes that can happen in the environment. In this paper, dynamic scheduling conditions are presumed, in which job parameters are not known before the jobs are released into the system and the schedule is constructed together with the execution of the system.

4 Evolving Dispatching Rules for Many-objective Optimisation Using Genetic Programming

This section will shortly describe the GP approach for generating new DRs. In order to generate DRs for several objectives simultaneously, four different multi-objective and many-objective algorithms will be used: NSGA-II [16], HaD-MOEA [69], MOEA/D [73] and NSGA-III [15]. NSGA-II was selected since it represents one of the most popular multi-objective algorithms, NSGA-III and MOEA/D since they represent two popular many-objective algorithms, while HaD-MOEA was selected since it was previously used for creating DRs for multi-objective scheduling problems. All algorithms use the same set of *crossover* and *mutation* operators. The following crossover operators are used: simple crossover, context-preserving crossover, size fair crossover, one point crossover and uniform crossover [65]. The operators are combined in a way that each time a crossover is performed, a randomly selected operator is applied. The set of mutation operators contains only one operator, the sub-tree mutation [65]. The maximum tree depth was set to 5 for all algorithms. In order to make the comparison between all algorithms as fair as possible, the maximum number of fitness evaluations was used as the termination condition for all the algorithms. The aforementioned algorithms were developed using the Evolutionary Computation Framework (ECF) [33].

4.1 The Scheduling Procedure

The DRs applied in this work consists of two parts: a meta-algorithm and a priority function. The role of the meta-algorithm is to determine which job should be processed on which machine. In order to make that decision, the meta-algorithm uses a certain priority function to determine the priorities of all the available jobs. Algorithm 1 represents the outline of the manually defined meta-algorithm, which assumes that an appropriate priority function has previously been evolved. The outlined algorithm finds the best appropriate mapping between a job and a machine. If the machine which was chosen for the job is available, then the job is scheduled on that machine. Otherwise, the scheduling of this job is postponed until a machine becomes available, or a new job enters the system. It should be emphasized that this meta-algorithm and the priority function are loosely coupled, meaning that the same meta-algorithm can be used with different priority functions, and vice versa.

Unlike the meta-algorithm, priority functions are not manually defined, but rather generated by using GP. In order for GP to be able to evolve an appropriate priority function, it must be able to use relevant information about jobs and machines, which describe the current state of the system. As a consequence, one of the most important preparatory steps is to carefully select the set of terminal nodes that may appear in the tree representation of the priority function. Table 1 represents the list of all terminal nodes used by GP. The *time* variable, which is used for defining some of the terminal nodes, represents the current time of the system.

In addition to the set of terminal nodes, GP also uses a set of functional nodes which combine the terminal nodes into a complete expression. For the experiments five operators were used in the set of functional nodes: $+$, $-$, $*$, $/$ and POS. The $/$ represents the protected division operator defined as $/(a,b) = \begin{cases} 1, & \text{if } |b| < 0.000001 \\ \frac{a}{b}, & \text{else} \end{cases}$,

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 π_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 π_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
```

Table 1: Terminal nodes

Node name	Description
pt	processing time of job j on the machine i (p_{ij})
pmin	the minimal job processing time on all machines: $\min\{p_{ij}\}\forall i$
pavg	the average processing time on all machines
PAT	patience - the amount of time until the machine with the minimal processing time for the current job will be available
MR	machine ready - the amount of time until the current machine becomes available
age	the time that the job spent in the system: $time - r_j$
dd	due date (d_j)
SL	positive slack: $max\{d_j - p_{ij} - time, 0\}$
w_t	tardiness weight (w_{Tj})
w_c	completion time weight (w_{Cj})
w_e	earliness weight (w_{Ej})

while POS is defined as $POS(a) = max\{a, 0\}$. This operator set was chosen since in a previous study it was shown that GP was able to achieve the best results by using such an operator set [19].

GP uses both terminal and functional nodes in order to design an expression tree which will denote the priority function which should be used by the meta-algorithm. Figure 1 represents an example of a possible GP expression tree. This specific expression tree would be decoded into a priority function which would look like: $w * pt + \frac{pt}{dd} * SL$.

4.2 Multi-objective Performance Measures

In order to be able to evaluate the quality of the obtained Pareto fronts, appropriate performance measures are used [38]. In this paper the following two measures will be used: *inverted generational distance* and *hypervolume*.

Inverted generational distance (IGD) [38] represents a diversity-convergence metric which measures the distance between the the approximated Pareto front P and

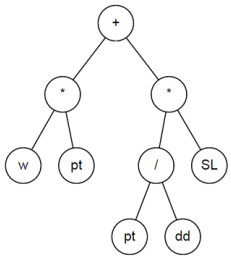


Fig. 1: Example of an expression tree generated by GP

Pareto front S obtained by some algorithm. Since it measures the distance between those two Pareto fronts, its value should be minimised. This metric is defined as:

$$IGD(P,S) = \frac{(\sum_{i=i}^{|P|} d_i^q)^{1/q}}{|P|}, \tag{14}$$

where $d_i = min_{s \in S} \left\| F(p_i) - F(s_i) \right\|, p_i \in P$, $q = 2$ and $\|\cdot\|$ is defined as the Euclidean distance measure. From the definition it is obvious that d_i represents the distance from a solution in the approximated Pareto front to the closest solution in the obtained Pareto front S.

Hypervolume (HV) [76], a diversity-convergence metric, measures the amount of volume in the objective-space, which is covered by a given Pareto front. This metric is defined as:

$$HV(S,R) = volume(\bigcup_{i=1}^{|S|} v_i) \tag{15}$$

where v_i represents the hypercube constructed between the point with index i and the reference point R. Since this metric measures the space covered by the obtained Pareto front, its value should be maximised.

5 Experiment design

5.1 Benchmark Setup and Evaluation

In order to gain insights into the performance of the tested algorithms, a set of 120 problem instances was generated based on the methods described in related references for various scheduling environments [45], [18], [35]. The detailed procedure of how the problem instances are generated is available on the project web site at [34].

The set of 120 problem instances was divided into the training and test sets, each containing half of the problem instances. Depending on the problem instance, the number of jobs can be 12, 25, 50 or 100, while the number of machines can be 3, 6 or 10. Problem instances of different properties were used in order to evolve DRs which would be applicable on scheduling problems with different characteristics. The training set was used by the GP in order to evolve DRs for some given criteria, after

which the effectiveness of these evolved DRs was measured using the test set. The total fitness of an individual for a specific criterion was calculated as a sum of values of that criterion for each problem instance. Since problem instances have significantly different characteristics, this can also cause that smaller problem instances have little influence in the total objective value. In order to prevent this from happening, all the objectives have been normalised so that different problem instances have similar influence in the total objective value. For example, for the Twt criterion the objective is normalised with the following expression: $f_i = \frac{\sum_{j=1}^{n} w_{T_j} T_j}{n \bar{w}_{T_j} \bar{p}}$ where n denotes the number of jobs in the problem instance, $\bar{w}$ the average weight of the jobs, $\bar{p}$ the average job processing duration and i the index of the current problem instance. The normalisation of other objectives will be skipped for sakes of brevity, but the expressions are similar to the one denoted for the Twt criterion.

As mentioned beforehand, the dynamic scheduling environment is considered in this paper. This means that in the simulation environment jobs and their characteristics are not available from the start of the system execution, but rather become available when the job is released into the system. Because of that the scheduling is being performed concurrently with the system execution and as soon as a job or machine becomes available it is used in order to determine which of the available jobs will be scheduled next on one of the available machines.

In order to obtain statistically significant results, each experiment was executed 30 times. All the reported results are based on the performance on the *test set* of scheduling problems, in the following way: for each run, the Pareto front of non-dominated solutions was recorded. These results are then evaluated on the test set, and the Pareto front of solutions on the test set is recorded as the final solution set for that run (since not all solutions which were non-dominated on the training set are also non-dominated on the test set). For that obtained Pareto front three metrics were calculated: HV (using an arbitrary reference point), IGD (where the Pareto fronts of all experiments with the same many-objective criteria were combined into a single Pareto front of non-dominated solutions) and the percentage of nondominated solutions (ND) in the population, which is defined as the quotient of the number of nondominated solutions in the population and the size of the population. The values of the HV metric have additionally been scaled to the interval $[0, 10]$ for each criteria individually, since the HV values usually end up being quite large.

5.2 Parameter Optimisation

Since the performance of GP may significantly depend on the parameter values, parameter optimisation plays an important role in GP. Therefore a significant amount of time was invested to optimise the parameters of the algorithms. For determining the optimal parameter values it was chosen to minimise the many-objective criterion (Twt, Ft, Nwt, C_{max}, T_{max}, F_{max}). In order to decide which parameter value is better, parameter value with the best average value of the HV metric on all 30 runs will be selected, since this metric measures the quality of solutions both in terms of diversity and convergence [38]. For the stopping criterion 100 000 function evaluations were chosen since a larger number of evaluations did not lead to any improvements in the results.

Table 2: Results for various population sizes

Algorithm	Metric	Population size				
		50	100	200	500	1000
MOEA/D	HV	**5.692**	5.393	5.681	5.536	5.576
HaD-MOEA	HV	4.956	5.55	5.999	7.0377	**7.272**
NSGA-II	HV	7.107	5.665	8.217	8.474	**8.839**
		NSGA-III population size				
		56	100	252	462	1287
NSGA-III	HV	6.386	5.703	7.634	8.308	**8.381**

Table 3: Results for various mutation probabilities

Algorithm	Metric	Mutation probability				
		0.1	0.3	0.5	0.7	0.9
MOEA/D	HV	5.564	5.553	**5.692**	5.679	5.687
HaD-MOEA	HV	7.396	7.492	7.272	**7.579**	7.486
NSGA-II	HV	**8.945**	8.929	8.839	8.536	8.833
NSGA-III	HV	8.287	8.367	8.381	8.718	**8.915**

Table 4: Results for various neighbourhood sizes

Algorithm	Metric	Neighbourhood size				
		2	3	5	10	20
MOEA/D	HV	**5.86**	5.848	5.708	5.692	5.617
HaD-MOEA	HV	7.534	7.477	**7.579**	7.447	7.404

Table 2 represents the average HV value of the 30 independent runs results achieved for different population sizes for all the algorithms. The NSGA-III algorithm uses different population sizes since it is suggested that the population size is more or less equal to the number of reference points [15]. An additional problem is that the number of reference points in NSGA-III will not always be the same since it depends on the number of optimised criteria. Nevertheless, the number of reference points (and consequently the population size) is kept as close as possible to the obtained optimal value.

Table 3 displays the average HV values based on the 30 independent runs, achieved for different mutation rates. For this parameter it can be seen that different algorithms prefer different values, unlike for the population size.

Since HaD-MOEA and MOEA/D use the neighbourhood size, an additional parameter which is not used by the other two algorithms, this parameter is optimised as well. Table 4 represents the average HV values obtained for different neighbourhood sizes. From the results it can be seen that both algorithms achieve better results for smaller neighbourhood sizes, MOEA/D for the size two and HaD-MOEA for the size of five.

The MOEA/D algorithm has an additional parameter, which determines how the decomposition of multi-objective optimisation is done. Three different decomposition methods are compared: the Tchebycheff approach, the normalized Tchebycheff approach and the boundary intersection (BI) approach. Based on the average HV

Table 5: Results for various decomposition methods

Algorithm	Metric	Mutation probability		
		Tchebycheff	norm. Tchebycheff	BI
MOEA/D	HV	5.86	**6.449**	4.318

values presented in Table 5, the normalized Tchebycheff decomposition method was chosen.

5.3 Many-objective criteria combinations

In order to properly analyse the performance of DRs generated for optimising multiple objectives, several different objective combinations will be considered. The criteria combinations were chosen in order to discern if specific criteria combinations can have an influence on the quality of generated DRs.

The criteria combinations will first be chosen to analyse how the different objective types influence the quality of generated DRs. The (Nwt, T_{max}, Twt) combination will be used in order to analyse if grouping only one type of criteria, namely due date related criteria, can lead to good performance of the generated DRs. Other criteria combinations will group two criteria types, like completion time and flowtime (such as in criteria combinations (C_{max}, F_{max}, Ft) and (C_{max}, C_w, M_{ut}, F_{max}, Ft)), due date and flowtime (like in combination (F_{max}, Ft, Nwt, T_{max}, Twt)), or due date and completion time (like in combination (C_{max}, Cw, Twt)). However, some combinations like (C_{max}, Cw, F_{max}, Ft, Nwt, T_{max}, Twt), are also considered in order to see how grouping of different combination types influences the results.

Additionally, it is also interesting to test whether grouping criteria of the same definitions can also lead to good results. For that purpose, criteria which are defined as *maximum values* will be grouped together in the criteria combination (C_{max}, F_{max}, T_{max}), as well as criteria which are defined as *weighted sums* in combinations (Cw, Ft, Twt) and (Cw, $Etwt$, Ft, Nwt, Twt)

Finally, several criteria combinations will also be tested in order to analyse how the inclusion and exclusion of the $Etwt$ and M_{ut} criteria influence the quality of the evolved DRs. The reason why the influence of these two criteria are separately analysed is because optimising them leads to extremely bad performance on most of the other scheduling criteria. Therefore it is interesting to see how the inclusion of such criteria, which are quite negatively correlated with the other criteria, can influence the quality of the generated DRs. For this analysis several criteria combinations of sizes five and seven which include those criteria (such as combinations (C_{max}, $Etwt$, F_{max}, Ft, M_{ut}, Nwt, T_{max}), (C_{max}, $Etwt$, F_{max}, Ft, M_{ut}, Nwt, T_{max}) and (C_{max}, $Etwt$, Ft,M_{ut}, Twt)), and those which do not include them (like (F_{max}, Ft, Nwt, T_{max}, Twt) and (C_{max}, Cw, F_{max}, Ft, Nwt, T_{max}, Twt)) will be used for evolving DRs.

6 Results and discussion

6.1 Automatic generation of DRs for optimising multiple objectives

This section will present the results of automatically designed DRs for many-objective optimisation, and the obtained results will be compared to those achieved by standard manually designed DRs and automatically designed DRs.

6.1.1 Comparison of results achieved by MOGP and SGP

The goal of this subsection is to analyse if by optimising several criteria simultaneously by using multi-objective GP (MOGP) algorithms, the automatically generated DRs will still be able to match the performance of DRs generated by the single objective GP (SGP). This comparison will be performed in a way that, from each of the 30 runs of MOGP algorithms, the best value for each criterion is extracted and the average, minimum and standard deviation metrics are calculated. In the tables, the best value for each result will be denoted in bold, while those values in which MOGP achieves a better result than SGP will be denoted in grey. Additionally, it will also be analysed how the different criteria combinations which are optimised also influence the ability of MOGP to outperform SGP.

Table 6 represents the results of MOGP and SGP algorithms for various criteria combinations when optimising three criteria simultaneously. From the results it can deduced that in most cases the many-objective algorithms are able to achieve the same performance as the single objective GP. For some criteria combinations it can even be seen that MOGP can generate DRs which significantly outperform DRs designed by SGP. The most evident example of this is when optimising the (Nwt, T_{max}, Twt) criteria combination, where NSGA-III was able to evolve DRs that outperform the best results of SGP by 3.5%, 5.6% and 5.5% for the Nwt, T_{max} and Twt criterion respectively. For the other criteria combinations the differences are not as prominent, but nevertheless the DRs evolved by MOGP algorithms were able to perform better or equally well as SGP, in most of the experiments. From the four tested algorithms, the NSGA-III algorithm has shown to outperform the results of SGP for all criteria combinations, while NSGA-II and HaD-MOEA have also shown to outperform SGP for most objectives. On the other hand, MOEA/D has shown to struggle more than the other algorithms, but still managed to outperform SGP for several objectives.

Table 7 represents the results of MOGP and SGP algorithms when optimising five criteria simultaneously. Again, it can be seen that the MOGP algorithms can achieve equally good results as the SGP algorithm, in most of the cases. For the (Cw, $Etwt$, Ft, Nwt, Twt) criteria combination the MOGP algorithms once again achieved quite good performance, even outperforming SGP by 13% for the $Etwt$ criterion. For the (F_{max}, Ft, Nwt, T_{max}, Twt) criteria combination the algorithms have also shown to perform quite well, outperforming the best results of SGP by 2.4% for the Twt criterion, 3.4% for the Nwt criterion, and performing equally well as SGP for the other three criteria. However, for this number of criteria it is also evident that the MOGP algorithms start to have problems in order to outperform results from SGP. Additionally, it can also be seen that the HaD-MOEA and MOEA/D algorithms have difficulties in performing as well as the SGP algorithm.

Table 6: Results for optimising three criteria

Metrics		Algorithm				
		MOEA/D	HaD-MOEA	NSGA-II	NSGA-II	SGP
		C_{max}, F_{max}, T_{max}				
C_{max}	avg	38.25	38.23	38.09	**38.08**	38.29
	stdev	0.11	0.142	**0.095**	0.1	0.15
	min	37.99	37.93	**37.83**	37.87	38.02
F_{max}	avg	14.15	14.1	13.88	**13.81**	13.95
	stdev	0.228	0.164	0.131	**0.11**	0.194
	min	13.72	13.81	13.63	**13.59**	13.6
T_{max}	avg	2.594	2.519	2.5	**2.447**	2.7
	stdev	0.118	0.096	0.088	**0.046**	0.291
	min	2.372	2.383	**2.371**	**2.371**	2.376
		Nwt, T_{max}, Twt				
Nwt	avg	6.677	6.572	6.468	**6.453**	7.005
	stdev	0.159	0.17	**0.135**	0.146	0.326
	min	6.422	6.235	6.275	**6.164**	6.384
T_{max}	avg	2.56	2.525	2.491	**2.419**	2.7
	stdev	0.093	0.09	0.082	**0.075**	0.291
	min	2.391	2.359	2.327	**2.25**	2.376
Twt	avg	13.83	13.6	13.3	**13.18**	13.66
	stdev	0.393	0.53	0.365	**0.327**	0.426
	min	13.21	12.94	12.72	**12.28**	12.96
		Cw, Ft, Twt				
Cw	avg	871.1	869.8	869.3	**869.1**	875.2
	stdev	1.509	0.824	0.545	**0.325**	0.872
	min	869.3	869	**867.9**	**867.9**	873.8
Ft	avg	154.8	154.4	154.2	**154**	155.3
	min	0.828	0.667	**0.529**	0.577	0.879
	stdev	153.4	153.3	153.4	**153.1**	153.9
Twt	avg	14.42	14.08	13.89	**13.63**	13.66
	stdev	0.629	0.53	0.534	0.462	**0.426**
	min	13.19	13.02	13.03	**12.79**	12.96
		C_{max}, F_{max}, Ft				
C_{max}	avg	38.24	38.17	38.13	**38.05**	38.29
	stdev	0.124	0.1	0.66	**0.087**	0.15
	min	**37.88**	37.95	37.98	37.89	38.02
F_{max}	avg	14.08	14.05	13.88	**13.79**	13.95
	min	0.159	0.242	0.133	**0.11**	0.194
	stdev	13.82	13.67	13.63	**13.59**	13.6
Ft	avg	155.2	154.3	154.2	**154.1**	155.3
	min	1.157	0.334	0.543	**0.341**	0.879
	stdev	153.9	153.5	**153.1**	153.5	153.9
		C_{max}, Cw, Twt				
C_{max}	avg	38.3	38.26	38.21	**38.14**	38.29
	stdev	0.12	0.143	**0.116**	0.137	0.15
	min	38.1	**37.87**	37.91	37.88	38.02
Cw	avg	871.2	869.7	**869.1**	869.2	875.2
	min	1.793	0.629	0.5	**0.289**	0.872
	stdev	868.6	868.9	**868**	868.6	873.8
Twt	avg	14.5	13.9	13.7	**13.37**	13.66
	stdev	0.443	0.373	**0.325**	0.374	0.426
	min	13.38	13.15	12.96	**12.7**	12.96

Table 7: Results for optimising five criteria

Metrics		Algorithm				
		MOEA/D	HaD-MOEA	NSGA-II	NSGA-III	SGP
Cw, *Etwt*, *Ft*, *Nwt*, *Twt*						
Cw	avg	875	872.2	**871.4**	871.6	875.2
	stdev	3.808	1.953	1.162	1.543	**0.872**
	min	869.3	869.2	869.3	**868.9**	873.8
Etwt	avg	307	323.5	**259.7**	273.12	274.2
	stdev	28.76	29.04	14.66	**13.83**	18.17
	min	261	279.8	**209.4**	242.7	236.9
Ft	avg	158.6	156.2	**154.8**	155.3	155.3
	stdev	4.506	1.723	**0.878**	0.96	0.879
	min	153.5	153.7	**153.4**	153.9	153.9
Nwt	avg	6.91	6.615	6.549	**6.461**	7.005
	stdev	0.188	0.135	0.125	**0.108**	0.321
	min	6.454	6.457	6.329	**6.27**	6.384
Twt	avg	14.27	13.7	13.39	**13.26**	13.66
	stdev	0.682	0.367	0.314	**0.288**	0.426
	min	13.12	13.19	12.86	**12.66**	12.96
C_{max}, C_w, M_{ut}, F_{max}, *Ft*						
C_{max}	avg	38.3	38.15	38.06	**38.04**	38.29
	stdev	0.103	0.118	0.108	**0.079**	0.15
	min	38.06	37.9	**37.83**	37.85	38.02
C_{max}	avg	872.9	870.5	869.8	**869.2**	875.2
	stdev	2.839	1.207	1.022	**0.443**	0.872
	min	868.3	869.4	868.5	**867.9**	873.8
M_{ut}	avg	0.058	0.057	**0.053**	**0.053**	**0.053**
	stdev	0.006	0.004	**0.002**	**0.002**	0.003
	min	0.048	0.049	0.047	**0.046**	**0.046**
F_{max}	avg	14.22	14.07	13.94	**13.9**	13.95
	stdev	0.228	0.174	0.16	**0.118**	0.194
	min	13.9	13.76	13.74	13.64	**13.6**
Ft	avg	156.8	154.6	154.1	**153.8**	155.3
	stdev	3.132	0.911	0.312	**0.299**	0.879
	min	153.8	153.4	153.7	**153.2**	153.9
F_{max}, *Ft*, *Nwt*, T_{max}, *Twt*						
F_{max}	avg	14.55	14.2	13.89	**13.84**	13.9
	stdev	0.437	0.359	0.161	0.109	0.194
	min	13.9	13.7	**13.6**	13.66	**13.6**
Ft	avg	155.3	154.2	**153.8**	**153.8**	155.3
	stdev	0.888	0.638	0.341	**0.331**	0.879
	min	153.9	153.4	153.3	**153.2**	153.9
Nwt	avg	6.737	6.646	6.408	**6.394**	7.005
	stdev	0.478	0.151	0.131	**0.119**	0.326
	min	6.342	6.365	6.174	**6.121**	6.384
T_{max}	avg	2.605	2.478	2.425	**2.41**	2.7
	stdev	0.12	0.064	0.06	**0.041**	0.291
	min	2.36	2.385	**2.33**	2.35	2.376
Twt	avg	14.06	13.74	13.18	**13.09**	13.66
	stdev	0.478	0.44	0.293	**0.224**	0.426
	min	13.32	12.8	**12.68**	12.72	12.96
C_{max}, *Etwt*, *Ft*,M_{ut}, *Twt*						
C_{max}	avg	38.52	38.16	**38.15**	38.27	38.29
	stdev	0.274	**0.106**	0.126	0.151	0.15
	min	38.09	37.9	**37.87**	38.01	38.02
Etwt	avg	311.9	316.2	**261.4**	273.2	274.2
	stdev	32.78	26.14	21.35	**13.13**	18.17
	min	267.7	269	**200**	239.9	236.9
Ft	avg	158.1	155.7	**154.8**	163.8	155.3
	stdev	1.094	1.229	0.986	6.2	**0.879**
	min	153.4	**153.3**	**153.3**	155.7	153.9
M_{ut}	avg	0.062	0.61	**0.051**	0.052	0.053
	stdev	0.007	0.004	**0.002**	**0.002**	0.003
	min	0.051	0.052	**0.046**	0.048	**0.046**
Twt	avg	15.42	13.8	13.72	14.09	**13.66**
	stdev	1.094	0.32	**0.313**	0.45	0.426
	min	13.67	13.01	13.16	13.42	**12.96**

Table 8 represents the results of MOGP and SGP algorithms when optimising five criteria simultaneously. For the (C_{max}, Cw, F_{max}, Ft, Nwt, T_{max}, Twt) combination, all algorithms have shown to generally outperform SGP, for most of the criteria. However, for the other two combinations, the MOGP algorithms are not able to outperform SGP for all of the criteria. For those criteria combinations, DRs evolved by NSGA-II are shown to outperform those of SGP in most cases, but the other three MOGP algorithms were unable to perform as well.

Table 9 represents the results of MOGP and SGP algorithms when optimising nine criteria. For this criteria combination it is even more evident that the MOGP algorithms exhibit problems to achieve better results than SGP. The NSGA-II algorithm has shown to achieve the best performance here, while the other three algorithms achieved better performance than SGP only for a few criteria.

The results have demonstrated that MOGP algorithms are able to outperform or at least perform equally well as SGP, for most of the tested criteria combinations. Although MOGP demonstrated better performance on smaller number of criteria than on larger ones, it nevertheless seems that it is the combination of optimised criteria that has a larger influence on the efficiency. For example, grouping similar criteria like in (Nwt, T_{max}, Twt) has shown to enable MOGP to significantly outperform SGP for each of the criteria. It seems that by grouping similar criteria together the MOGP algorithms have a "wider" look on the problem, which allows them to achieve much better performance on each criteria. However, when different criteria types are mixed with each other, it can be seen that the performance of MOGP algorithms depends on which types of criteria are combined. For example, combining flowtime and completion time criteria did not influence the effectiveness of MOGP, but when combining the due date related criteria with the completion time criteria as in (C_{max}, Cw, Twt), or when combining different types of criteria as in (Cw, Ft, Twt), it can be seen that the performance of MOGP algorithms decreases. However, it is interesting to note that as the number of criteria increases, but they still contain different criteria types as in the (F_{max}, Ft, Nwt, T_{max}, Twt) and (C_{max}, Cw, F_{max}, Ft, Nwt, T_{max}, Twt) combinations, the MOGP algorithms are able to outperform SGP in most of the cases. It is also interesting to note that the performance of some MOGPs can even increase than when optimising only three different criteria types. Therefore it is beneficial to include several criteria of the same type in the combination, since this can lead to better performance than when only using a single criteria of each type. Combining criteria of the same definitions did not show any significant influence on the results, therefore it can be concluded that grouping criteria in this way is not beneficial.

It is interesting to analyse further how the inclusion of the $Etwt$ and M_{ut} criteria influences the performance of MOGP algorithms. When optimising five criteria, it can be seen that by including only one of those two criteria the MOGP algorithm are still able to outperform SGP. However, MOEA/D and HaD-MOEA algorithms have more problems when these criteria are included. When both of the criteria are included, then even the performance of NSGA-III drops, and only NSGA-II is able to outperform the results of SGP almost consistently. When optimising seven criteria a similar thing can be observed. NSGA-II is usually the algorithm which is able to outperform results of SGP almost for all objectives. On the other hand, the other three algorithms struggle much more, especially if both the $Etwt$ and M_{ut} are included. When optimising nine criteria, even NSGA-II starts to struggle to outperform SGP. Based on the previous observations, it is evident that for a smaller

Table 8: Results for optimising seven criteria

Metrics		Algorithm				
		MOEA/D	HaD-MOEA	NSGA-II	NSGA-III	SGP
C_{max}, Cw, Ft, F_{max}, M_{ut}, T_{max}, Twt						
C_{max}	avg	38.35	38.16	**38.14**	38.2	38.29
	stdev	0.152	**0.112**	0.121	0.126	0.15
	min	38.07	**37.84**	37.86	37.89	38.02
Cw	avg	874.4	872.1	**870.5**	873.3	875.2
	stdev	2.703	2.274	0.99	2.481	0.872
	min	**868.8**	869.2	869.2	869.2	873.8
Ft	avg	157.2	154.6	**154.2**	155.4	155.3
	stdev	3.363	0.86	**0.555**	1.85	0.879
	min	153.5	**153.3**	153.5	153.6	153.9
F_{max}	avg	14.62	14.08	14.04	14.13	**13.95**
	stdev	0.523	**0.148**	0.165	0.23	0.194
	min	13.91	13.85	13.65	13.76	**13.6**
M_{ut}	avg	0.066	0.6	**0.052**	0.053	0.53
	stdev	0.007	0.005	**0.002**	**0.002**	**0.002**
	min	0.054	0.047	**0.046**	0.047	**0.046**
T_{max}	avg	2.647	2.48	**2.43**	2.505	2.7
	stdev	0.164	0.083	**0.042**	0.075	0.291
	min	2.446	2.377	**2.359**	2.412	2.376
Twt	avg	14.51	13.83	**13.54**	14.255	13.66
	stdev	0.8	**0.388**	0.45	0.389	0.426
	min	13.24	13.01	**12.85**	13.75	12.96
C_{max}, $Etwt$, F_{max}, Ft, M_{ut}, Nwt, T_{max}						
C_{max}	avg	38.37	**38.18**	38.2	38.36	38.29
	stdev	0.209	**0.074**	0.086	0.112	0.15
	min	38	38.04	**37.97**	38.18	38.02
$Etwt$	avg	308.5	321.8	**264.5**	303.8	274.2
	stdev	38.14	29.14	**17.74**	27.15	18.17
	min	217.6	270.4	**213**	224.8	236.9
F_{max}	avg	14.44	14.2	14.05	14.56	**13.95**
	stdev	0.479	0.254	**0.13**	0.42	0.194
	min	13.88	13.83	13.78	13.97	**13.6**
Ft	avg	159.6	156.8	**155.2**	160.433	155.3
	stdev	5.648	2.041	1.072	5.171	**0.879**
	min	154.1	153.8	153.8	**153.5**	153.9
M_{ut}	avg	0.064	0.061	**0.051**	0.052	0.053
	stdev	0.007	0.004	**0.002**	0.003	0.003
	min	0.052	0.049	0.047	**0.046**	**0.046**
Nwt	avg	7.04	**6.703**	6.704	6.856	7.005
	stdev	0.212	**0.133**	0.139	0.151	0.321
	min	6.545	**6.357**	6.407	6.492	6.384
T_{max}	avg	2.717	**2.417**	**2.417**	2.475	2.7
	stdev	0.173	**0.038**	0.068	0.064	0.291
	min	2.427	2.362	**2.188**	2.344	2.376
C_{max}, Cw, F_{max}, Ft, Nwt, T_{max}, Twt						
C_{max}	avg	38.29	38.25	**38.11**	38.13	38.29
	stdev	0.084	0.103	**0.103**	0.112	0.15
	min	38.07	38.02	**37.88**	37.96	38.02
Cw	avg	871.4	870.6	**869.1**	869.6	875.2
	stdev	1.533	1.622	**0.264**	0.409	0.872
	min	869.3	868.2	**868.4**	868.7	873.8
F_{max}	avg	14.25	14.21	13.92	**13.88**	13.95
	stdev	0.289	0.251	0.195	**0.112**	0.194
	min	13.75	13.7	**13.5**	13.64	13.6
Ft	avg	154.3	154.1	**153.6**	153.9	155.3
	stdev	0.136	0.512	**0.265**	0.367	0.879
	min	153.5	153.5	**153**	153.2	154.4
Nwt	avg	6.837	6.582	6.439	**6.419**	7.005
	stdev	0.136	0.178	0.141	**0.112**	0.321
	min	6.557	6.163	**6.112**	6.164	6.384
T_{max}	avg	2.613	2.473	2.424	**2.416**	2.7
	stdev	0.097	0.073	0.066	**0.043**	0.291
	min	2.44	2.376	**2.352**	2.357	2.376
Twt	avg	14.37	13.56	13.21	**13.16**	13.66
	stdev	0.417	0.381	0.371	**0.338**	0.426
	min	13.44	12.83	**12.58**	12.77	12.96

Table 9: Results for optimising nine criteria

Metrics		Algorithm				
		MOEA/D	HaD-MOEA	NSGA-II	NSGA-III	SGP
C_{max}	avg	38.35	38.2	**38.18**	38.3	38.29
	stdev	0.172	**0.077**	0.112	0.111	0.15
	min	**37.89**	38	37.98	38.09	38.02
Cw	avg	875	874.5	**872.7**	879.1	875.2
	stdev	3.9	3.02	2.072	3.251	**0.872**
	min	870.5	870.4	**869.8**	874.2	873.8
$Etwt$	avg	321	327.3	279.4	289.7	**274.2**
	stdev	26.11	29.69	17.32	**14.04**	18.17
	min	275.9	282.5	253.7	270.3	**236.9**
F_{max}	avg	14.69	14.27	14.05	14.48	**13.95**
	stdev	0.748	0.27	**0.142**	0.421	0.194
	min	14.03	13.89	13.76	13.9	**13.6**
Ft	avg	159.5	157.1	155.6	161	**155.3**
	stdev	4.475	2.638	1.252	4.231	**0.879**
	min	154.3	154.1	**153.7**	154.4	153.9
M_{ut}	avg	0.065	0.061	**0.051**	0.053	0.053
	stdev	0.008	0.006	**0.002**	0.003	0.003
	min	0.052	0.048	**0.046**	0.0482	**0.046**
Nwt	avg	6.924	6.684	**6.677**	6.804	7.005
	stdev	0.198	0.148	**0.134**	0.152	0.321
	min	6.536	6.414	6.434	6.531	**6.384**
T_{max}	avg	2.637	**2.4**	2.428	2.444	2.7
	stdev	0.156	0.051	0.059	**0.048**	0.291
	min	2.389	2.28	**2.269**	2.325	2.376
Twt	avg	14.73	13.67	13.71	14.15	**13.66**
	stdev	0.915	0.388	**0.359**	0.37	0.426
	min	13.39	12.97	13.08	13.48	**12.96**

number of criteria including either $Etwt$ or M_{ut} will not cause big problems, and that most algorithms will still be able to outperform SGP. However, by increasing the number of criteria or including both criteria in the combination, the results of all algorithms start to deteriorate and are more and more unable to outperform results of SGP.

Based on the results shown in this section, it can be concluded that MOGP algorithms are for most criteria combinations able to perform better, or equally well as SGP. If criteria of equal type are grouped together and optimised, the MOGP algorithms are even able to achieve significantly better performance than SGP. As for the dependence on the criteria combinations, it was shown that the performance of MOGP algorithms depends more on the combination of criteria which are optimised, than on their number. MOGP algorithms performed well for optimising all criteria combinations which did not include the $Etwt$ and M_{ut} criteria, since optimising those two leads to a bad performance on all other criteria, and therefore the size of the Pareto front largely increases.

6.1.2 Analysis of the influence of the generated DRs

The results in the previous section give a good overview of the performance of different many-objective algorithms, and how good solutions they can achieve for the different criteria when compared to SGP. However, with these results alone it is impossible to determine the actual quality of the developed DRs. Therefore, the goal of this section is to compare several automatically generated DRs for optimising multiple objectives with certain standard manually designed DRs.

In this section the performances of several generated many-objective DRs will be compared to the results achieved by four popular DRs for the unrelated machines environment (min-min [14], max-min [8], sufferage [25], min-max [32]). The criterion for selecting the generated DRs was that they achieve better results for as many optimisation criteria as possible, when compared to manually defined DRs. Table 10 represents the results achieved by the various DRs. The row denoted with GA represents results which were achieved by using a genetic algorithm (GA), while the row denoted by SGP shows the very best results achieved by the single objective GP when it was applied to each of the criteria independently. This means that the results denoted in this row were not achieved by a single DR, but rather nine DRs, each evolved for a different criterion. Naturally, DRs are not expected to generate schedules of equal quality as the GA (since GA performs scheduling in static conditions, while DRs solve it in dynamic conditions). However, GA results give a good estimate of the lower bounds for this problem. For the many-objective DRs only the results for the criteria for which they were evolved are denoted in the table, while the rest of them will be marked with "-". In addition, for the generated DRs each objective for which the DR achieved better results than all of the manually defined DRs will be shown in bold.

First the DRs for the three objectives will be analysed. Here it can be seen that rules $R1$, $R2$ and $R3$ achieve better values for each of the criteria for which they were optimised, than any of the manually defined DRs. For rule $R1$ it is also interesting to note that it achieved even better results than the best DRs which were evolved by SGP for the three optimised criteria. This demonstrates that many-objective algorithms can outperform single objective algorithms, if the criteria which should be optimised are wisely chosen, such as in this case where all the optimised criteria were due date related. On the other hand, rules $R4$ and $R5$ were not able to outperform the best results of all standard DRs. The problems in this case seem to have occurred because the C_{max} criterion was included in the optimisation set. This just demonstrates that even for a smaller number of objectives the algorithms can struggle depending on the criteria combination being optimised. However, although rule $R6$ also contains the C_{max} criterion it achieved a worse result only for C_{max}, and to such an extent which could be considered almost negligible (0.1% worse than the sufferage DR). Therefore, it seems that the C_{max}, F_{max} and T_{max} criteria are much more suitable to be optimised together than the previous two combinations.

By analysing DRs generated for five criteria it can be seen that the generated DRs are usually able to outperform the standard DRs in four criteria, which can be seen from rules $R7$, $R9$ and $R10$. Rules $R9$ and $R10$ optimise the same criteria combination, however here it can be seen how DRs were unable to optimise both the F_{max} and Ft criteria, and therefore had to make a compromise between two of them. For the other two criteria combinations the many-objective algorithms were unable to generate DRs which could optimise them all equally well. This can be seen

Table 10: Comparison of different DRs

method	Criteria								
	C_{max}	Cw	$Etwt$	F_{max}	Ft	M_{ut}	Nwt	T_{max}	Twt
GA	36.79	844.4	80.55	12.74	140.8	0.006	5.340	1.897	9.533
SGP	38.02	873.8	236.9	13.60	154.0	0.046	6.384	2.376	12.96
Manually defined DRs									
min-min	38.31	877.7	997.8	15.66	157.2	0.129	7.144	3.161	16.72
max-min	38.84	916.9	968.6	14.27	195.9	0.127	8.138	3.107	22.07
sufferage	37.93	881.7	992.9	14.68	161.0	0.123	7.195	3.054	16.65
min-max	38.07	887.8	988.2	14.25	167.3	0.129	7.793	3.019	17.49
Evolved DRs - three objectives									
$R1$	-	-	-	-	-	-	**6.326**	**2.310**	**12.86**
$R2$	-	-	-	-	-	-	**6.566**	**2.249**	**12.28**
$R3$	-	**871.8**	-	-	**153.9**	-	-	-	**15.01**
$R4$	38.49	**876.3**	-	-	-	-	-	-	**13.62**
$R5$	**37.89**	-	-	**13.61**	175.5	-	-	-	-
$R6$	37.97	-	-	**13.62**	-	-	-	**2.754**	-
Evolved DRs - five objectives									
$R7$	-	**877.5**	992.5	-	**157.1**	-	**6.734**	-	**14.46**
$R8$	38.36	**872.7**	-	15.80	**155.9**	0.126	-	-	-
$R9$	-	-	-	15.80	**157.2**	-	**6.620**	**2.545**	**13.78**
$R10$	-	-	-	**13.95**	165.7	-	**6.941**	**2.436**	**14.73**
$R11$	38.49	-	**961.1**	-	189.2	**0.123**	-	-	**14.29**
Evolved DRs - seven objectives									
$R12$	38.39	**876.4**	-	15.88	**156.8**	**0.123**	-	**2.683**	**14.64**
$R13$	38.47	-	**961.1**	15.78	189.5	**0.123**	**6.975**	**2.429**	-
$R14$	38.57	-	**965.5**	**14.25**	200.5	**0.122**	8.276	**2.965**	-
$R15$	38.36	**877.0**	-	15.42	**156.5**	-	**6.699**	**2.692**	**14.28**
Evolved DRs - nine objectives									
$R16$	38.71	909.7	**961.7**	16.49	187.7	**0.123**	**6.991**	**2.517**	**14.21**
$R17$	38.60	**876.6**	995.4	15.72	**155.4**	0.137	**7.097**	**2.831**	**14.82**

from rules $R8$ and $R11$ which outperform the best results from standard DRs for only two and three objectives respectively. Both of these criteria combinations had the $Etwt$ and M_{ut} criteria included, which leads to the conclusion that it is hard to simultaneously optimise these criteria within the given criteria combinations.

When optimising seven criteria it can be seen that the algorithms usually manage to generate DRs which outperform the standard DRs in four or five objectives, depending on the combination which is optimised. It can be seen that the rules mostly focus on optimising due date and flowtime related criteria. It is interesting to note that none of the presented DRs were able to outperform the standard DRs in the C_{max} criterion, however the algorithms did evolve DRs which were able to do so but at the expense of greatly deteriorating the other criteria.

Finally, rules $R16$ and $R17$ represent two examples of DRs which were evolved for the many-objective problem consisting of all nine criteria. In most cases the algorithms generated DRs which focused on optimising the due date related objectives (Twt, Nwt and T_{max}) and either on the $Etwt$ and M_{ut} criteria (as rule $R16$), or on Cw and Ft criteria (as rule $R17$). It is interesting to note that the expression of

rule $R17$ is of similar size as that of rule $R1$, which demonstrates that the number of objectives does not directly influence the size of the evolved DR.

Although automatically generated DRs have shown to achieve better results than standard DRs in most cases, this does nevertheless mean that automatically designed DRs are superior to standard DRs in all cases. This certainly depends on the criteria set which needs to be optimised, and can be seen from rule $R8$. For this criteria combination it was shown that no DR could perform better than the standard DRs for more than two criteria. Therefore, in such a case it does not make sense to develop new DRs, since they can not outperform the standard DRs in an extent that would justify the effort and time for their development. Additionally, it can also be seen that for most of the DRs denoted in table 10, the DRs were unable to outperform the standard DRs for the C_{max} criterion. Therefore, if this criterion is one of the more important ones, it would also probably prove to be more worthwhile to use a standard DR than to develop a multi-objective one, since the results again are not in the favour of automatically designed DRs.

An additional thing which can be seen from the results in table 10 is that GP is able to evolve DRs which can outperform standard DRs in at most five criteria. Therefore, even if more than five criteria are optimised simultaneously, GP will probably be unable to evolve a DR which could outperform the standard DRs for all of them. Nevertheless, optimising more than five criteria can also prove useful in order to find out for which criteria the many-objective GP can evolve good DRs, and for which it will struggle.

In order to demonstrate how the evolved DRs look like, we give examples of rules $R1$:

$$pmin+PAT+\frac{pmin}{w_T}-(pt+MR+pavg*w_T)+\frac{SL}{MR*w_T^2}+dd+w_T+pmin-\frac{w}{age}-w_T,$$

and $R17$:

$$((PAT+w_C)*(w_T-pavg)-(w_E+pavg+w+dd))*$$
$$((PAT+pmin)*(w_C-dd)-(w_C-dd-(pt+MR))-(\frac{dd}{pt}-pt).$$

It should be stressed out that the job which achieves the smallest priority value will be selected. After a short analysis of rule $R1$, several important building blocks of the rule can be outlined. Since the DR is optimised for due date criteria, one of the most important parts of the rule is $\frac{SL}{MR*w_T^2}+dd$. The dd part of the expression denotes that jobs which have a larger due date will have a larger priority value. On the other hand, the $\frac{SL}{MR*w_T^2}$ expression will have a larger value for jobs which have larger slack values (amount of time until they would be late), and have a smaller priority. From the rest of the DR it was shown that the expression $PAT+\frac{pmin}{w_T}-pt-MR$ is also important. The sub-expression $\frac{pmin}{w_T}$ will prioritise jobs which have a smaller minimum execution time and larger priority, while the PAT terminal gives more importance to jobs whose best machine will be free soon. The other part of the expression, $-pt-MR$ prefers jobs which have a larger processing time and machines which will be free in a much latter time, however it seems that this part of the DR is important since it will allow for the rule to schedule jobs on machines other than their best machine. The other expressions have shown to be useless in the rule and seem to represent noise which was accumulated during the evolution process, since

with their removal, the fitness of the rule can even slightly improve. Therefore, rule $R1$ can be reduced to the following rule, without any deterioration in performance:

$$PAT + \frac{pmin}{w_T} - pt - MR + \frac{SL}{MR * w_T^2} + dd.$$

The second rule is extremely hard to analyse in the given form, however, after manual simplification, rule $R17$ can be written as:

$$PAT{*}pavg{*}dd{*}(PAT{+}pmin){-}pavg{*}PAT{*}(pt{+}MR){+}dd{*}dd{*}pmin{-}dd{*}(pt{+}MR),$$

without big influence in its overall quality. Although the rule is still quite complicated, parts of it can now be analysed. For example the $PAT{*}pavg{*}dd{*}(PAT{+}pmin)$ will have a larger value for jobs with a later due date, but also that have longer processing times and whose machine on which they achieve their minimum processing time is taken the longest. The expression $dd * dd * (PAT + pmin)$ functions similarly. The other two expressions prefer jobs with longer processing times and machines which become ready later in time. This is a bit surprising, but probably these two expressions serve to distribute the jobs more evenly on all the machines, since their value is usually smaller than of the other two previously denoted expressions. Therefore, generally this rule will prefer jobs which have a close due date, and which can quickly be executed.

Based on the previous analysis it can be seen that the two rules are quite hard to interpret in their original form, however, when simplified it can be seen that some general conclusions can be drawn from them. The interpretability of rules could probably be improved by using dimensionally aware GP [41] which should generate more interpretable expressions. Many tree simplification techniques have also shown to not only effectively decrease the sizes of generated expressions, but also evaluation time of expressions and their interpretability. These methods are can be based on algebraic simplification rules [72], numerical simplification [42], or numerical relaxation of algebraic rules [39]. Similarly, weights which constitute the degree of contribution could also be introduced for each node and a gradient descent method could be used in order to determine the weights of each node, and therefore also the importance of each node and subtree [71]. Another way of controlling the size could be to include the size of the DRs as an additional objective in the optimisation process, however it is questionable how this would reflect on the overall algorithm performances.

The results presented in this section have shown that automatically generated multi-objective DRs can perform quite well when compared to standard manually designed DRs. However, there is a limit to the number of criteria in which the multi-objective DRs can outperform standard DRs, since it was shown that generated multi-objective DRs can not outperform standard DRs in more than five criteria. Nevertheless, automatically designed multi-objective DRs offer greater flexibility since they allow for the choice of objectives for which the DR should perform best, thus allowing generation of DRs tailored for specific situations and problems.

6.2 Correlation of scheduling criteria

In the previous section it was shown that the performance of algorithms and the quality of DRs depends highly on the criteria set which was optimised. This section

will analyse how the different scheduling criteria are correlated with each other, and how optimising one criteria has an influence on optimising other criteria.

In order to visualise the interdependencies between different criteria, all results of individual algorithms were aggregated into a single Pareto front of nondominated solutions, for the case of optimising all nine criteria. Figure 2 represents the visualisation of Pareto fronts for all pairs of two different objectives. Based on the results in the previous section it is possible to observe that certain objectives are nonconflicting, and therefore the many-objective algorithms were able to find DRs which perform well on all of them. This is most evident for the due date related criteria (Nwt, T_{max}, Twt), for which the algorithms achieved very good performance and evolved high quality DRs. The Pareto fronts of the $Twt - T_{max}$ and $T_{max} - Nwt$ criteria combinations show that the solutions form almost a straight line for those combinations, meaning that by optimising one criterion, the other criterion will also be indirectly optimised. In addition to those three criteria, the Ft and Cw criteria have also shown a positive correlation between each other, which can be seen not only from Figure 2, but also from the fact that many DRs denoted in the last section usually achieved good results simultaneously for both of the criteria. Many-objective DRs have also shown to perform well when the (C_{max}, F_{max}, T_{max}) objective is optimised. It was also shown that by optimising due date related criteria it was possible to optimise the Ft and Cw criteria to a certain degree. This demonstrates that even different kind of criteria can be positively correlated and that it is possible to mutually optimise them. A reason for this is possibly due to the fact that all those criteria try to schedule the jobs as soon as possible, the difference is just depending on whether the jobs will be scheduled as to finish as fast as possible, to remain as less as possible in the system, or to finish before their due date.

On the other hand, several criteria have proven to be quite hard for many-objective GP to optimise simultaneously with others. One such example is the C_{max} criterion, which could usually be optimised well together with the F_{max} criterion, but only at the expense of decreasing the performance of the DRs on other criteria. A more interesting relation can be observed between the Cw and C_{max}, Ft and F_{max} criteria. Although the first two are completion time related, while the second two are flowtime related, it was shown that the algorithms were unable to evolve good DRs which could optimise both. This is demonstrated by optimising the (C_{max}, Cw, F_{max}, Ft, M_{ut}) combination, where the algorithms were able to evolve DRs which can outperform the standard DRs in only two out of the five criteria. One such rule was $R8$ which was denoted in table 10, and which was shown to be able to optimise only the Cw and Ft criteria, while for the others it achieved inferior results. Other rules evolved for this criteria set would optimise C_{max} and F_{max} criteria, but never any combination of those with either the Cw of Ft criteria. Therefore, unlike for the due date related criteria, where the different criteria have shown to be positively correlated, this was not shown to be the case for the flowtime or completion time related ones.

There is no apparent reason why for the due date related criteria it is possible to simultaneously optimise the Twt and T_{max}, and why for C_{max} and C_w, or Ft and F_{max} this is not the case. One possible explanation could be that the due date related criteria are more flexible, meaning that not all jobs will in the end contribute to the criteria value, rather only the jobs which are late will do so, and such jobs represent only a fraction of jobs in the environment. On the other hand, there is no such flexibility for the other two criteria types, since every job could in the end

directly contribute to the value of the criteria. Therefore it is much harder to make compromises since, for example, trying to minimise the maximum flowtime could have a negative effect on several other jobs and thus increase the total flowtime of the entire system.

Although many of the previously considered criteria are negatively correlated, optimising one such criterion will not lead to extremely bad results on the other criteria. Unfortunately, this can not be said for the $Etwt$ and M_{ut} criteria, by whose optimisation the results for the other criteria deteriorate drastically. This is a consequence of the way they are defined, since for example the $Etwt$ criterion will try to complete all the jobs as close to their due date as possible. This will naturally have a negative effect on the flowtime criteria, since jobs will stay longer in the system, or the tardiness criteria, since it will be needed to compromise between the earliness and tardiness of jobs, and also on completion time, since jobs will not be finished as soon as possible, but rather as close to their due date which can be in a much later time. Therefore, in many objective optimisation these two criteria could be seen as additional secondary objectives, rather than the main criteria which need to be optimised as best as possible, since this would lead to inferior results on all the other criteria in the optimisation set.

From the observations in this section it can be concluded that MOGP algorithms have shown to perform better if positively correlated criteria are optimised together. However, even optimising negatively correlated criteria is not extensively problematic, if by optimising one criteria, the values of the other are not extremely deteriorated, like for Twt when optimising C_{max}. More problematic are criteria like $Etwt$ and M_{ut} where by optimising those criteria, the values for the other ones become extremely bad. Therefore, great care needs to be taken when choosing the set of criteria which needs to be optimised, since the combination of criteria will heavily influence the quality of the results of MOGP algorithms.

6.3 Performance analysis of MOGP algorithms

This section will analyse the performances of all algorithms for different criteria number and combinations, with regards to the convergence and diversity of the obtained Pareto fronts. The goal of this section is to compare the different MOGP algorithms with each other, and provide an analysis on why each of them performs well on a given situation.

For each criteria combination the average values of the HV, IGD and ND metrics are calculated and presented in tables. The Wilcoxon rank-sum test is also used to determine if there are significant differences for the HV and IGD metrics between the individual algorithms. The differences are considered significant if the obtained value for p is smaller than 0.05. The distributions of the aggregated Pareto fronts for each algorithm and each criteria they optimised will also be compared, and to visualise these distributions the box-plot representations will be used.

Table 11 presents the results achieved for optimising various combinations of three criteria, while Figure 3 represents the box-plot figures of the HV and IGD metrics. For the HV metric NSGA-II and NSGA-III achieved significantly better results for all criteria combinations when compared to the other two algorithms. As for the IGD criteria, NSGA-III and NSGA-II have again proven to be significantly

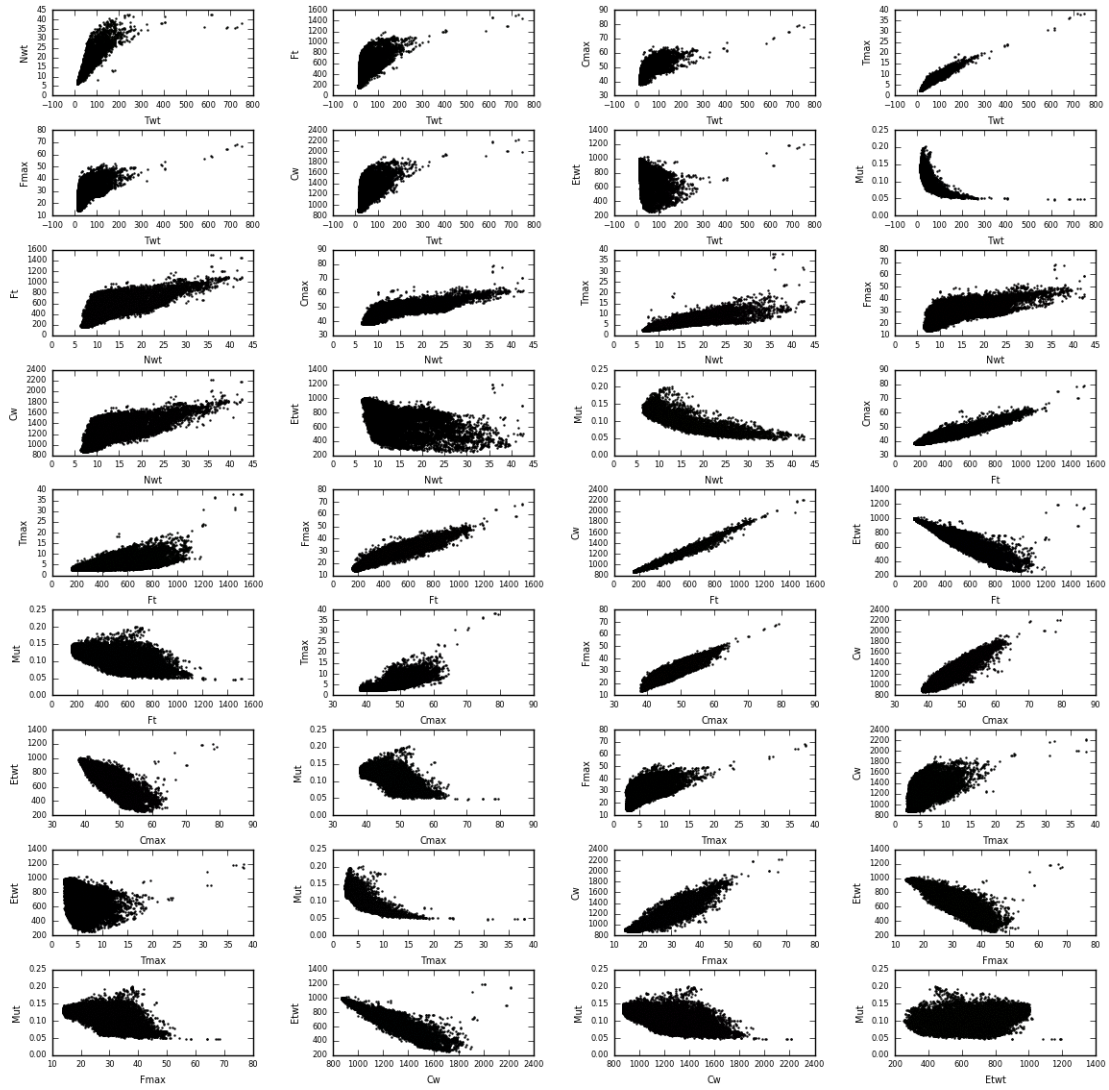


Fig. 2: Pareto fronts for pairwise objective combinations when optimising nine criteria

better than the other two algorithms, with NSGA-III achieving significantly better results than NSGA-II for all criteria combinations except the (Nwt, T_{max}, Twt).

The solution distributions for three criteria are shown in Figure 4. The red dots in the box-plots denote the maximum outlier, while the purple dots denote minimum outliers. In most cases the solution distributions are quite similar between the algorithms (like for optimising the (C_{max}, C_w, Twt) criteria combination), or algorithms tend to have a good distribution on one or two criteria, while on the others it is not as good (for example NSGA-III for the (C_{max}, F_{max}, T_{max}) criteria combination).

Table 11: HV and IGD values for optimising three criteria

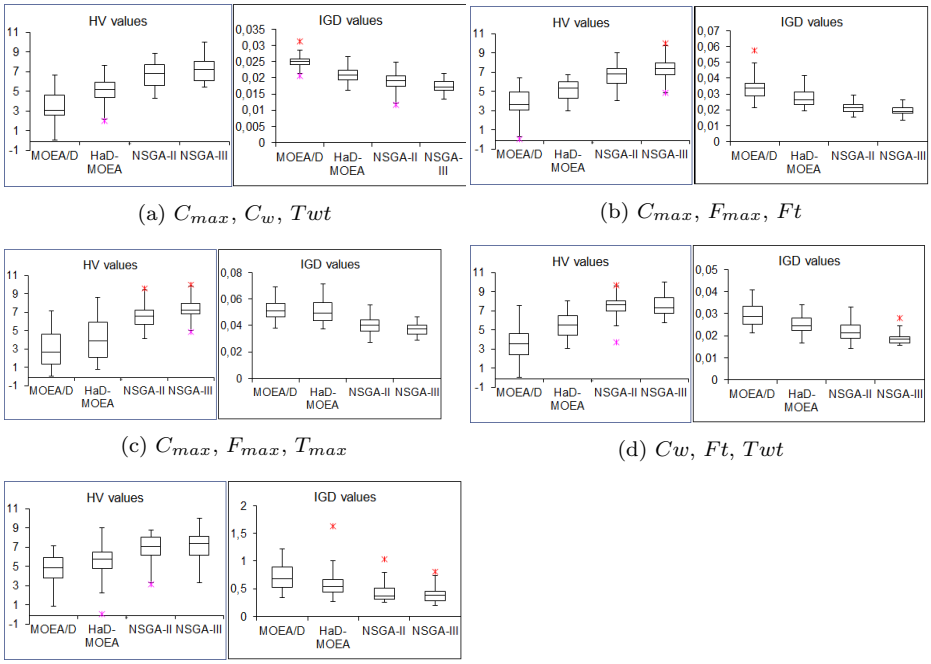
Metrics	Algorithm			
	MOEA/D	HaD-MOEA	NSGA-II	NSGA-III
C_{max}, F_{max}, T_{max}				
HV	3.036	4.006	6.649	**7.309**
IGD	0.05221	0.05015	0.04009	**0.03682**
ND	**0.4**	0.2	0.3	0.23
Nwt, T_{max}, Twt				
HV	4.698	5.706	6.872	**7.25**
IGD	0.70834	0.58894	0.44280	**0.38160**
ND	**5**	0.35	0.45	0.31
Cw, Ft, Twt				
HV	3.523	5.539	7.4	**7.578**
IGD	0.02965	0.02525	0.02184	**0.01891**
ND	**15**	1.5	3.25	2.1
C_{max}, F_{max}, Ft				
HV	3.84	5.031	6.661	**7.341**
IGD	0.03363	0.02798	0.02145	**0.019514**
ND	**15**	1.7	3.1	2.2
C_{max}, Cw, Twt				
HV	3.312	5.04	6.651	**7.201**
IGD	0.02522	0.02100	0.01906	**0.01751**
ND	**15.4**	2.1	3.8	2.7

The only situation where a considerable gap between the results exists is for optimising the (Nwt, T_{max}, Twt) criteria combination, where the NSGA-III algorithm achieved the best solution distributions for all three criteria.

Table 12 presents the results for optimising several combinations of five criteria. The box-plot representations for the HV and IGD values can be seen in figure 5. For the HV metric, NSGA-II achieves significantly better results than all other algorithms for all criteria combinations, except for the (C_{max}, Cw, M_{ut}, F_{max}, Ft) and (F_{max}, Ft, Nwt, T_{max}, Twt) where there is no significant difference between it and NSGA-III. On the other hand, for the IGD criteria, the NSGA-III algorithm achieved significantly better results for all criteria combinations, except for (C_{max}, $Etwt$, Ft, M_{ut}, Twt) and (F_{max}, Ft, Nwt, T_{max}, Twt), where there was no significant difference between it and NSGA-II, and for (C_w, $Etwt$, Ft, Nwt, Twt) where NSGA-II achieved better results.

Figure 6 represents the solution distributions of the aggregated Pareto fronts for each algorithm separately. It can be seen that for three combinations, namely (Cw, $Etwt$, Ft, $Nwt\ Twt$), (C_{max}, Cw, M_{ut}, F_{max}, Ft) and (C_{max}, $Etwt$, Ft, M_{ut}, Twt), a single algorithm cannot provide good distributions over all criteria. For the (F_{max}, Ft, Nwt, T_{max}, Twt) combination and for the due date related criteria all algorithms deliver quite similar solution distributions, while for the other two criteria the NSGA-II algorithm performs worse than the other algorithms (which is due to the quite large outlier values for the Ft and F_{max}).

Table 13 presents the results achieved for three combinations of seven criteria, while the box-plot representation of these results are shown in Figure 7. For the HV



(a) C_{max}, C_w, Twt

(b) C_{max}, F_{max}, Ft

(c) C_{max}, F_{max}, T_{max}

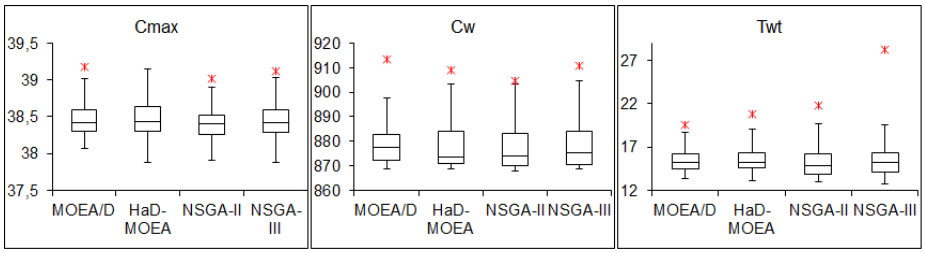
(d) Cw, Ft, Twt

(e) Nwt, T_{max}, Twt

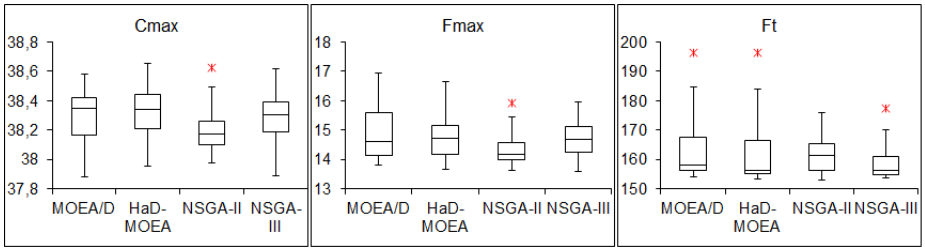
Fig. 3: HV and IGD values for optimising three criteria

Table 12: HV and IGD values for optimising five criteria

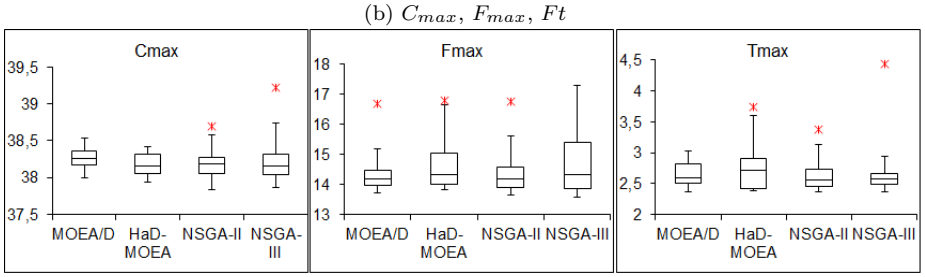
Metrics	Algorithm			
	MOEA/D	HaD-MOEA	NSGA-II	NSGA-III
Cw, $Etwt$, Ft, Nwt, Twt				
HV	1.871	3.558	**8.453**	7.61
IGD	0.00210	0.00156	**0.00080**	0.00087
ND	**65.6**	24	52.3	52.2
C_{max}, C_w, M_{ut}, F_{max}, Ft				
HV	2.926	6.274	8.787	**8.958**
IGD	0.00244	0.00157	0.00124	**0.00112**
ND	**36.7**	8.1	18.8	20.2
F_{max}, Ft, Nwt, T_{max}, Twt				
HV	2.54	4.599	7.582	**7.735**
IGD	0.01018	0.00790	0.00524	**0.00495**
ND	**27.9**	4.4	17.5	11.7
C_{max}, $Etwt$, Ft,M_{ut}, Twt				
HV	2.733	5.276	**8.078**	7.478
IGD	0.00191	0.00120	0.00077	**0.00075**
ND	**72.6**	35.7	40.9	47.5



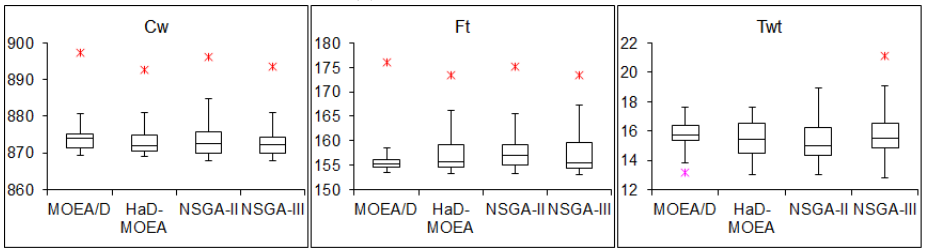
(a) C_{max}, C_w, Twt



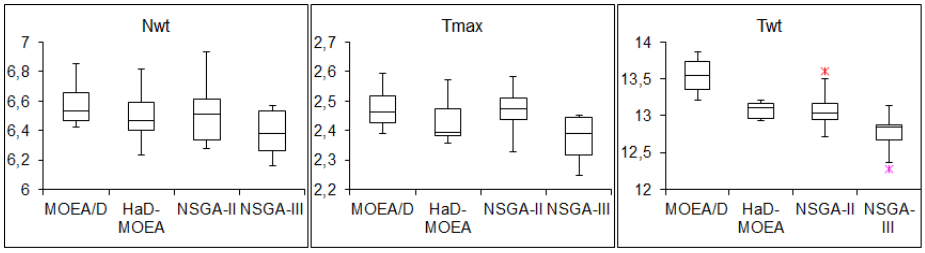
(b) C_{max}, F_{max}, Ft



(c) C_{max}, F_{max}, T_{max}

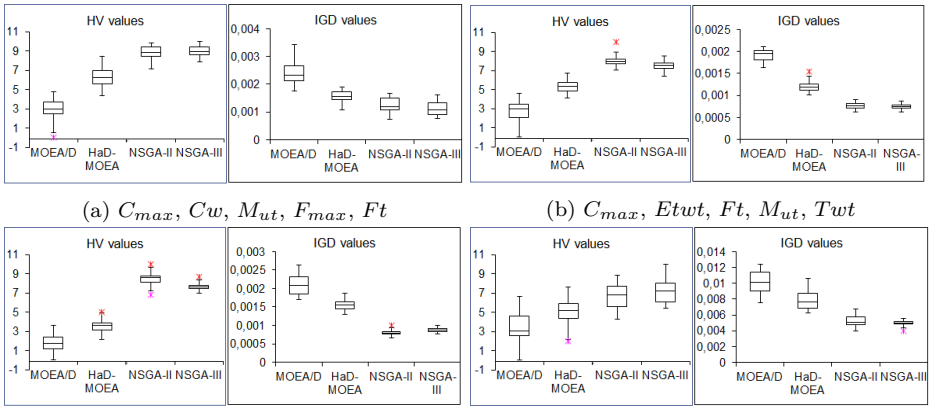


(d) Cw, Ft, Twt



(e) Nwt, T_{max}, Twt

Fig. 4: Solution distributions for three criteria



(a) C_{max}, Cw, M_{ut}, F_{max}, Ft

(b) C_{max}, $Etwt$, Ft, M_{ut}, Twt

(c) Cw, $Etwt$, Ft, Nwt Twt

(d) F_{max}, Ft, Nwt, T_{max}, Twt

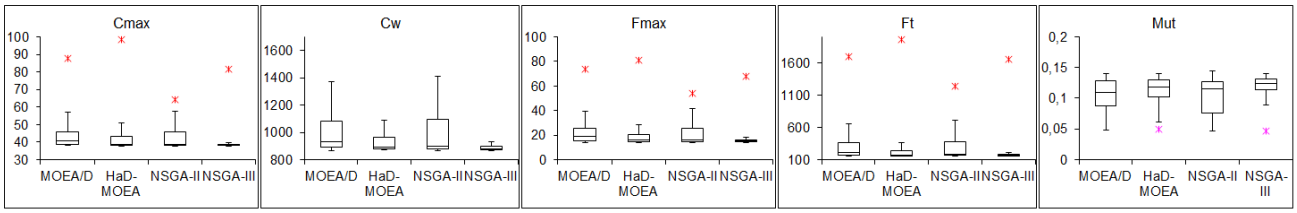
Fig. 5: HV and IGD values for optimising five criteria

Table 13: HV and IGD values for optimising seven criteria

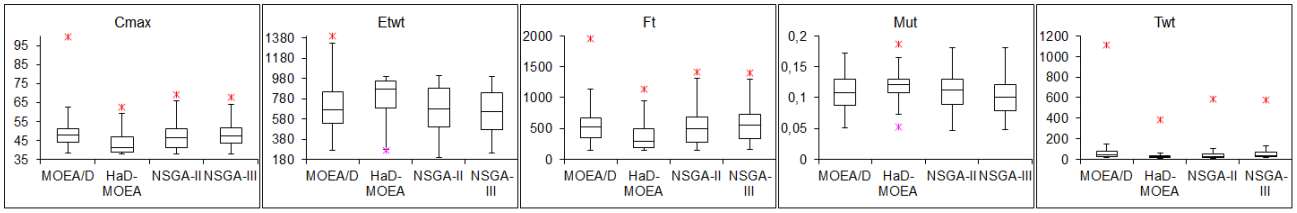
Metrics	Algorithm			
	MOEA/D	HaD-MOEA	NSGA-II	NSGA-III
C_{max}, Cw, Ft, F_{max}, M_{ut}, T_{max}, Twt				
HV	3.541	7.196	**9.1**	8.1
IGD	0.00406	0.00257	**0.00149**	0.001543
ND	**39.5**	13.2	24.9	24.8
C_{max}, $Etwt$, F_{max}, Ft, M_{ut}, Nwt, T_{max}				
HV	2.949	6.674	**8.756**	7.773
IGD	0.00174	0.00116	**0.00081**	0.00085
ND	**77.9**	44.2	59.5	66.8
C_{max}, Cw, F_{max}, Ft, Nwt, T_{max}, Twt				
HV	1.55	4.449	7.561	**7.592**
IGD	0.00767	0.00613	**0.00432**	0.00450
ND	**43.1**	7.7	23.7	24.3

metric the NSGA-II algorithm achieved significantly better results than any other algorithm for the criteria combinations in figures 7a and 7c, while for the combination in figure 7b there was no significant difference between it and NSGA-III. For the IGD metric, the NSGA-II algorithm has shown to achieve significantly better results than any other algorithm, except for the combination in figure 7c, where it did not achieve significantly better results than NSGA-III.

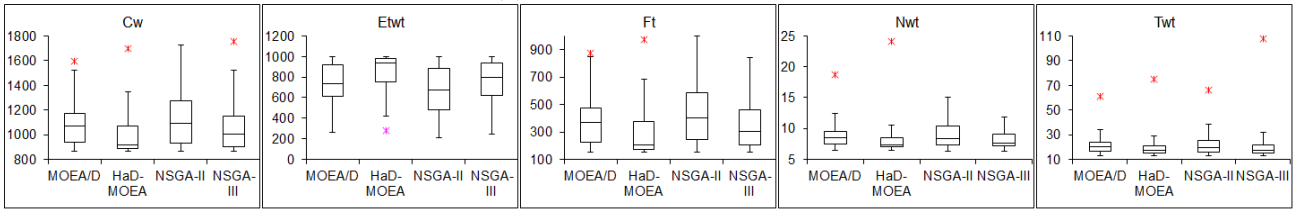
The distribution of solutions found by the algorithms is presented in figure 8. Here it can again be seen how neither of the algorithms achieves the best distributions for all criteria. Once again the $Etwt$ and M_{ut} criteria have proven to be problematic. Regarding the solution distributions, it is hard to proclaim any algorithm as the best, since the distributions are quite similar and depend heavily on the given criteria, but the NSGA-II algorithm seems to have been able to achieve the best distributions overall.



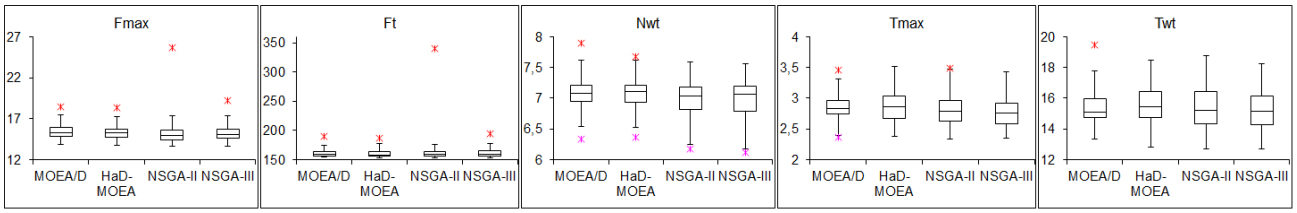
(a) C_{max}, Cw, M_{ut}, F_{max}, Ft



(b) C_{max}, $Etwt$, Ft, M_{ut}, Twt



(c) Cw, $Etwt$, Ft, Nwt Twt

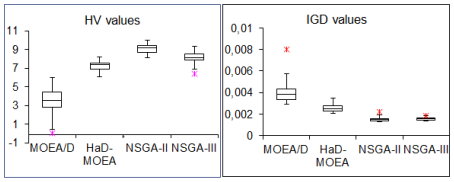


(d) F_{max}, Ft, Nwt, T_{max}, Twt

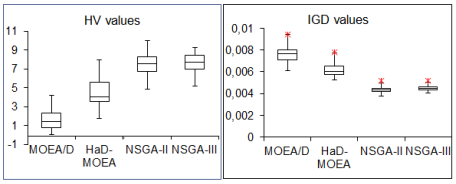
Fig. 6: Solution distributions for five criteria

Table 14 represents the results achieved for all nine criteria, while Figure 9 represents the distributions of HV and IGD in box-plot presentation. For this criteria combination, the NSGA-III algorithm achieves significantly better results than other algorithms, for both the HV and IGD metrics. Therefore, even if it was unable to find the best minimum values for the scheduling criteria, it was able to obtain the most diverse Pareto front out of all the algorithms. Figure 10 represents the solution distributions for all algorithms. Again it can be seen that no algorithm is able to dominate in distributions for all criteria.

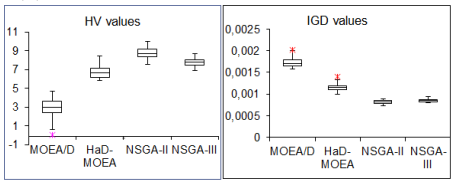
In the experiments it was shown that no single algorithm was able to achieve the best results over all of the tested criteria subsets. For the smallest combination sizes of three, the best results were usually achieved by the NSGA-III algorithm, which



(a) C_{max}, Cw, Ft, F_{max}, M_{ut}, T_{max}, Twt

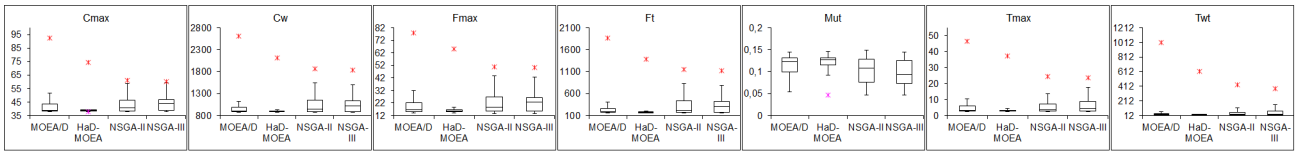


(b) C_{max}, Cw, Ft, F_{max}, Nwt, T_{max}, Twt

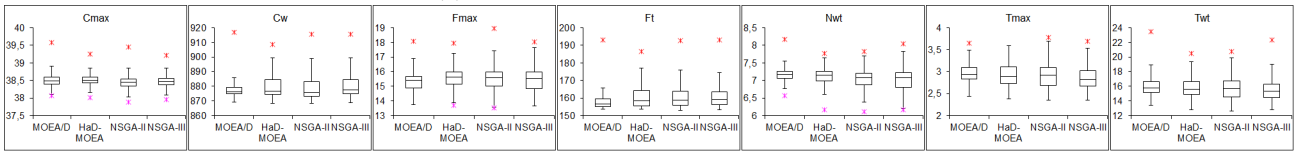


(c) C_{max}, $Etwt$, F_{max}, Ft, M_{ut}, Nwt, T_{max}

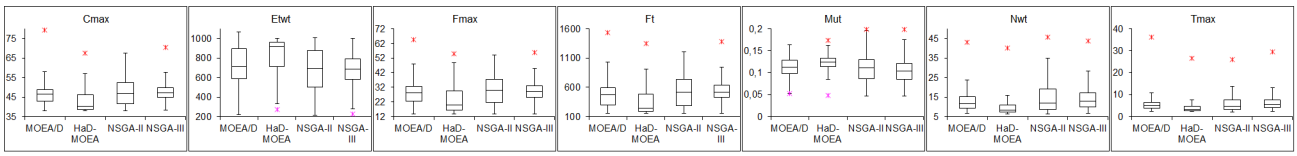
Fig. 7: HV and IGD values for optimising seven criteria



(a) C_{max}, Cw, Ft, F_{max}, M_{ut}, T_{max}, Twt



(b) C_{max}, Cw, Ft, F_{max}, Nwt, T_{max}, Twt



(c) C_{max}, $Etwt$, F_{max}, Ft, M_{ut}, Nwt, T_{max}

Fig. 8: Solution distributions for seven criteria

Table 14: HV and IGD values for optimising nine criteria

Metrics	Algorithm			
	MOEA/D	HaD-MOEA	NSGA-II	NSGA-III
HV	2.565	6.497	7.872	**8.179**
IGD	0.00157	0.00104	0.00077	**0.00068**
ND	**81.3**	44.9	58.2	63.3

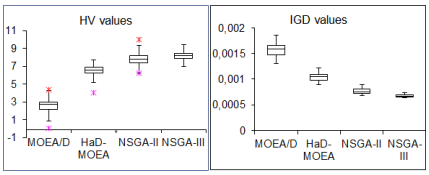


Fig. 9: HV and IGD values for optimising nine criteria

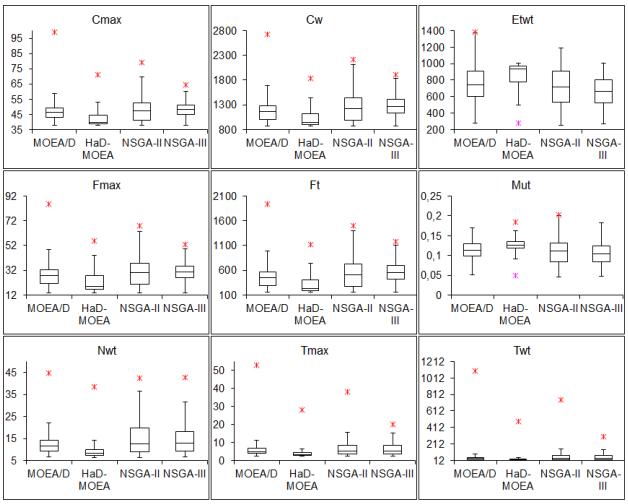


Fig. 10: Solution distributions for nine criteria

can be seen not only from the HV and IGD values, but also through the fact that it achieved the best values for most of the optimised criteria. However, as the number of criteria grew it was possible to observe that the NSGA-III started to encounter difficulties, as it was sometimes outperformed by NSGA-II. For optimising five criteria, NSGA-III still achieves better results than NSGA-II for most of the criteria (except for combinations where both the $Etwt$ and M_{ut} criteria are included). However, based on the HV and IGD metrics, there was mostly no difference between the two algorithms. Similar behaviour is also evident with seven criteria, where NSGA-III started achieving worse results than NSGA-II for most of the objectives. For the HV and IGD metrics the NSGA-II algorithm even outperformed NSGA-III for two out of the three tested criteria combinations. In the case of nine criteria, NSGA-III was once again able to outperform NSGA-II.

Such behaviour is quite surprising, since it would be expected that the NSGA-III algorithm performs better than NSGA-II for larger criteria combinations. The reason for this seems to originate from the way in which the solutions are selected into the next generation by the two algorithms. The reference point selection mechanism present in NSGA-III seems to favour convergence more than it does diversification. The reason for this seems to originate from the fact that more reference points will be present at the center of the Pareto front, and therefore the algorithm will simply select more solutions which are close to the center. On the other hand, NSGA-II

will select those solutions which are in areas that are less populated, which will most likely be on the edges of the Pareto fronts. As a consequence, NSGA-III will perform well on smaller criteria combinations, where the Pareto front is not large, and therefore the algorithm will be able easily converge to good solutions. Additionally, NSGA-III also seems to handle larger criteria combinations well, as long as the criteria which are included in those combinations are not negatively correlated, where optimising one criterion would lead to bad values for other criteria (such as the inclusion of $Etwt$ and M_{ut}), since in such a situation the Pareto front becomes large, and there is more need for diversification. The solution distributions also back-up these observations, since for several criteria NSGA-III has shown to obtain less distributed solutions than NSGA-II, like for example (C_{max}, C_w, M_{ut}, F_{max}, Ft), (Cw, $Etwt$, Ft, Nwt, Twt), (C_{max}, $Etwt$, F_{max}, Ft, M_{ut}, Nwt, T_{max}) and when optimising all nine criteria simultaneously. However, this is not the first time that it was shown that NSGA-II outperforms NSGA-III for a large number of criteria. Similar observations were also noticed in papers which dealt with comparison of many-objective algorithms on different types of problems [31] [30]. In those papers it was shown that NSGA-III performed quite poor for several different types of problems, and that NSGA-II was able to outperform it when optimising larger sets of criteria. The papers also suggest that the newly suggested many-optimisation algorithms (such as MOEA/D and NSGA-III) use selection mechanisms which are well suited for standard problems which are used to test the performance of many-objective algorithms, however when tested on different problems these algorithms can exhibit performance which is inferior to that of NSGA-II.

HaD-MOEA has mostly achieved results which are inferior to the results of NSGA-II and NSGA-III. This algorithm works in a similar fashion as NSGA-II, with the only difference being the calculation of the crowding distance, where it is calculated as the harmonic distance of n closest neighbours of the current point. However, by using such a crowding distance measure, HaD-MOEA has not only achieved a very small percentage of nondominated solutions in each run. Additionally, when optimising criteria combinations which included the $Etwt$ and M_{ut} criteria in it, from the solution distributions it can be seen that HaD-MOEA has shown to focus mostly on the other criteria, and this achieved quite bad solution distributions for the $Etwt$ and M_{ut} criteria. The cause for this problem is probably due to the fact that the closest neighbours are used to calculate the crowding distance. This crowding distance can perform badly if small groups of solutions, which are far apart from each other, exist in the search space. The problem here is that since the crowding distance is calculated only on the nearest neighbours of a solution, it is possible that all solutions will have a similar value of the crowded distance, and therefore all the solutions could be deleted even if they are far away from other solutions. This can have a negative effect on diversity, which is especially important when optimising criteria combinations with large Pareto fronts.

The MOEA/D algorithm has shown to achieve quite poor performances through most of the optimised criteria. We believe that the reasons for such behaviour are twofold. The first reason is the same as for NSGA-III, meaning that the way the algorithm was designed is not appropriate for optimising the tested criteria sets. The second reason is probably due to the way in which the parameters were optimised, meaning that first the generic parameters were optimised, after which the algorithm specific parameters were optimised. It is possible that the combination method which is used in MOEA/D has a great influence on all other parameters, and therefore it is

possible that the optimal parameters for that combination method would differ from the ones which were determined in this paper. Also, since MOEA/D uses a quite small population compared to the other algorithms, it will be much more difficult for it to obtain a good set of solutions, especially for problems with large Pareto fronts.

Based on the results it can be concluded that there is much difference in the performance of the different MOGP algorithms. Because of its strong convergence ability, NSGA-III has shown to perform well on smaller number of criteria, and for larger combinations in which there are no criteria which are have a strong negative correlation with other criteria (like $Etwt$ and M_{ut}). For such cases NSGA-II has shown to perform much better, since out of all the tested algorithms it produced more diverse solutions, and is therefore better able to cover the Pareto front. The other two MOGP algorithms, although perform well on certain objective combinations, were in most cases inferior to NSGA-II and NSGA-III.

7 Conclusion

This article analysed the performance of DRs generated by four multi-objective and many-objective algorithms for evolving dispatching rules for different many-objective scheduling problems. The generated DRs have proven to be not only competitive to some standards DRs on many of the tested criteria subsets, but have also shown to be able to outperform them in many occasions, especially for smaller criteria sets. For such smaller sets it was also shown that the DRs generated by many-objective DRs could also outperform the results of the single objective DRs designed by standard GP. Their additional strength lies also in the fact that they are able to generate good schedules for criteria combinations for which standard DRs might not achieve good results. However it was noticed that on some criteria subset the algorithms struggled to evolve DRs which could really perform well over all the optimised criteria. Therefore, even if the algorithms have shown good performance, it is still necessary to be careful for which criteria set the optimisation is performed, as to increase the performance of the developed DRs as much as possible.

By analysing the performance of the different MOGP algorithms which were tested in this paper, several interesting conclusions can be drawn. It was shown that more than the number of optimised cirteira, the combination of criteria had a much larger impact on the achieved results, with algorithms performing much better if correlated criteria are optimised together. In order to further analyse the interdependence of criteria, the Pareto fronts for all pairs of criteria were plotted, and based on those images it was concluded that the $Etwt$ and M_{ut} criteria correlate negatively with all of the other objectives.

Finally, through the analysis of the tested MOGP algorithms it was concluded that the best performances were achieved by either NSGA-II or NSGA-III. NSGA-III has shown to have a much larger convergence ability than the other algorithms, which makes it more appropriate for smaller criteria combinations and combinations in which the optimised criteria are not extremely negatively correlated. On the other hand, NSGA-II has shown to achieve a much better diversification of solutions, which is beneficial in situations where the set of optimised criteria contains many negatively correlated criteria, and therefore a good diversification is needed in order to cover the entire Pareto front. MOEA/D and HaD-MOEA were in most cases unable to achieve

equally good results as the previous two algorithms. For HaD-MOEA the reason is mostly due to the choice of the crowding distance calculation method, which doesn't seem to perform well, while on the other hand for MOEA/D the reason seems to be due to the sequence in which the parameters were optimised.

In future studies the analysis will be extended to some other multi-objective and many-objective algorithms like: strength Pareto evolutionary algorithm 2 (SPEA2) [75]; many-objective metaheuristic based on the $R2$ indicator (MOMBI) [22], [26]; and Unified NSGA-III (U-NSGA-III) [66]. Additionally, the objective set will also be expanded as to include more non-standard criteria, which were not extensively covered in the literature up until now and for which standard DRs do not even exist. Additional topics which will also be studied are the use of different local search and similar heuristics which could lead to further improvement in the results and could also speed up the evolution process, as well as the use of related GP methods like GEP for evolving many-objective DRs. Finally, the automatic simplification of rules and more detailed analysis of the developed many-objective rules will also be studied in the future.

References

1. Allahverdi, A., Gupta, J.N.D., Aldowaisan, T.: A review of scheduling research involving setup considerations. Omega **27**(2), 219–239 (1999). DOI 10.1016/S0305-0483(98)00042-5
2. Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y.: A survey of scheduling problems with setup times or costs. European Journal of Operational Research **187**(3), 985–1032 (2008). DOI 10.1016/j.ejor.2006.06.060
3. Attar, S.F., Mohammadi, M., Tavakkoli-Moghaddam, R.: Hybrid flexible flowshop scheduling problem with unrelated parallel machines and limited waiting times. International Journal of Advanced Manufacturing Technology **68**(5-8), 1583–1599 (2013). DOI 10.1007/s00170-013-4956-3
4. Baykasoğlu, A., Özbakır, L.: Discovering task assignment rules for assembly line balancing via genetic programming. The International Journal of Advanced Manufacturing Technology **76**(1), 417–434 (2015)
5. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated design of production scheduling heuristics: A review. IEEE Transactions on Evolutionary Computation **20**(1), 110–124 (2016). DOI 10.1109/TEVC.2015.2429314
6. Branke, J., Pickardt, C.W.: Evolutionary search for difficult problem instances to support the design of job shop dispatching rules. European Journal of Operational Research **212**(1), 22–32 (2011). DOI 10.1016/j.ejor.2011.01.044
7. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., Freund, R.F., Boloni, L.L.: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing **61**(6), 810–837 (2001). DOI 10.1006/jpdc.2000.1714
8. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed computing **61**(6), 810–837 (2001)
9. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: A Classification of Hyper-heuristics Approaches. Handbook of Metaheuristics **57**, 449–468 (2010). DOI doi:10.1007/978-1-4419-1665-5_15
10. Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: Exploring Hyper-heuristic Methodologies with Genetic Programming. Computational Intelligence **1**, 177–201 (2009). DOI doi:10.1007/978-3-642-01799-5_6
11. Cardoen, B., Demeulemeester, E., Beliën, J.: Optimizing a multiple objective surgical case sequencing problem. International Journal of Production Economics **119**(2), 354 – 366 (2009)

12. Cheng, V.H.L., Crawford, L.S., Menon, P.K.: Air traffic control using genetic search techniques. Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328) **1**, 249–254 (1999). DOI 10.1109/CCA.1999.806209
13. Costa, a., Cappadonna, F.a., Fichera, S.: A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times. International Journal of Advanced Manufacturing Technology **69**(9-12), 2799–2817 (2013). DOI 10.1007/s00170-013-5221-5
14. Davis, E., Jaffe, J.M.: Algorithms for scheduling tasks on unrelated processors. Journal of the ACM (JACM) **28**(4), 721–736 (1981)
15. Deb, K., Jain, H.: An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints. Ieeexplore.Ieee.Org **18**(c), 1–1 (2013). DOI 10.1109/TEVC.2013.2281534
16. 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). DOI 10.1109/4235.996017
17. Dimopoulos, C., Zalzala, A.: Investigating the use of genetic programming for a classic one-machine scheduling problem. Advances in Engineering Software **32**(6), 489–498 (2001)
18. Dimopoulos, C., Zalzala, A.M.S.: A genetic programming heuristic for the one-machine total tardiness problem. Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999 **3**(1), 2207–2214 (1999). DOI 10.1109/CEC.1999.785549
19. Đurasević, M., Jakobović, D., Knežević, K.: Adaptive scheduling on unrelated machines with genetic programming. Applied Soft Computing **48**, 419 – 430 (2016). DOI http://dx.doi.org/10.1016/j.asoc.2016.07.025
20. Ferreira, C.: Gene Expression Programming : A New Adaptive Algorithm for Solving Problems. Complex Systems **13**(2), 1–22 (2001)
21. Fowler, L., Pfund, M., Yu, L., Fowler, J.W., Carlyle, W.M.: Development Of A Robust Scheduling Rule For A Printed Wiring Board Drilling Operation With Multiple Scheduling Objectives And Fixed Order Release / Pickup Times
22. Gomez, R.H., Coello, C.a.C.: MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator. 2013 IEEE Congress on Evolutionary Computation, CEC 2013 (1), 2488–2495 (2013). DOI 10.1109/CEC.2013.6557868
23. Hansen, J.V.: Genetic search methods in air traffic control. Computers and Operations Research **31**(3), 445–459 (2004). DOI 10.1016/S0305-0548(02)00228-9
24. Hart, E., Ross, P., Corne, D.: Evolutionary scheduling: A review. Genetic Programming and Evolvable Machines **6**(2), 191–220 (2005). DOI 10.1007/s10710-005-7580-7
25. Hensgen, D., Freund, R.F.: Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of Distributed Computing, Special Issue on software support for distributed computing **59**(2) (1999)
26. Hern, R., Coello, C.a.C.: MOMBI : A New Metaheuristic for Many-Objective Optimization Based on the R2 Indicator. Science (1), 2488–2495 (2013). DOI 10.1109/CEC.2013.6557868
27. Hildebrandt, T., Heger, J., Scholz-Reiter, B.: Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation pp. 257–264 (2010). DOI doi:10.1145/1830483.1830530
28. Hunt, R., Johnston, M., Hunt, R., Johnston, M.: Evolving " Less-myopic " Scheduling Rules for Dynamic Job Shop Scheduling with Genetic Programming pp. 927–934
29. Hunt, R., Johnston, M., Zhang, M.: Evolving Machine-Specific Dispatching Rules for a Two-Machine Job Shop using Genetic Programming (2014). DOI 10.1109/CEC.2014.6900655
30. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 3045–3052 (2016). DOI 10.1109/CEC.2016.7744174
31. Ishibuchi, H., Setoguchi, Y., Masuda, H., Nojima, Y.: Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. IEEE Transactions on Evolutionary Computation **PP**(99), 1–1 (2016). DOI 10.1109/TEVC.2016.2587749
32. Izakian, H., Abraham, A., Snasel, V.: Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments. In: Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on, vol. 1, pp. 8–12. IEEE (2009)
33. Jakobović, D.: Evolutionary computation framework. URL http://gp.zemris.fer.hr/ecf

34. Jakobović, D.: Project site. URL http://gp.zemris.fer.hr/scheduling/
35. Jakobović, D., Budin, L.: Dynamic scheduling with genetic programming. Genetic Programming (2006)
36. Jakobović, D., Jelenković, L., Budin, L.: Genetic Programming Heuristics for Multiple Machine Scheduling. Proceedings of the 10th European Conference on Genetic Programming **4445**, 321–330 (2007). DOI doi:10.1007/978-3-540-71605-1_30
37. Jakobović, D., Marasović, K.: Evolving priority scheduling heuristics with genetic programming. Applied Soft Computing Journal **12**(9), 2781–2789 (2012). DOI 10.1016/j.asoc.2012.03.065
38. Jiang, S., Ong, Y.S., Zhang, J., Feng, L.: Consistencies and contradictions of performance metrics in multiobjective optimization. IEEE Transactions on Cybernetics **44**(12), 2391–2404 (2014). DOI 10.1109/TCYB.2014.2307319
39. Johnston, M., Liddle, T., Zhang, M.: A Relaxed Approach to Simplification in Genetic Programming, pp. 110–121. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
40. Kaban, a.K., Othman, Z., Rohmah, D.S.: Comparison of dispatching rules in job-shop Schedulingproblem Usingsimulation: A case study. International Journal of Simulation Modelling **11**(3), 129–140 (2012). DOI 10.2507/IJSIMM11(3)2.201
41. Keijzer, M., Babovic, V.: Dimensionally Aware Genetic Programming. Proceedings of the Genetic and Evolutionary Computation Conference **2**, 1069–1076 (1999)
42. Kinzett, D., Johnston, M., Zhang, M.: Numerical simplification for bloat control and analysis of building blocks in genetic programming. Evolutionary Intelligence **2**(4), 151 (2009). DOI 10.1007/s12065-009-0029-9. URL http://dx.doi.org/10.1007/s12065-009-0029-9
43. Kolahan, F., Kayvanfar, V.: A heuristic algorithm approach for scheduling of multi-criteria unrelated parallel machines. In: World Academy of Science, Engineering and Technology (2009)
44. Koza, J.R.: Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines **11**(3-4), 251–284 (2010). DOI 10.1007/s10710-010-9112-3
45. Lee, Y.H., Bhaskaran, K., Pinedo, M.: A heuristic to minimize the total weighted tardiness with sequence-dependent setups. IIE Transactions **29**(1), 45–52 (1997). DOI 10.1080/07408179708966311
46. Lin, Y.K., Fowler, J.W., Pfund, M.E.: Multiple-objective heuristics for scheduling unrelated parallel machines. European Journal of Operational Research **227**(2), 239–253 (2013)
47. Maheswaran, M., Ali, S., Siegal, H., Hensgen, D., Freund, R.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99) (June 1999) (1999). DOI 10.1109/HCW.1999.765094
48. Masood, A., Mei, Y., Chen, G., Zhang, M.: Many-objective genetic programming for job-shop scheduling. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 209–216 (2016)
49. Miyashita, K.: Job-shop scheduling with genetic programming. Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2000) pp. 505–512 (2000)
50. Nguyen, S., Zhang, M., Johnston, M.: A sequential genetic programming method to learn forward construction heuristics for order acceptance and scheduling. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1824–1831 (2014). DOI 10.1109/CEC.2014.6900347
51. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. 2012 IEEE Congress on Evolutionary Computation, CEC 2012 (i), 10–15 (2012). DOI 10.1109/CEC.2012.6252968
52. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. IEEE Transactions on Evolutionary Computation **17**(5), 621–639 (2013). DOI 10.1109/TEVC.2012.2227326
53. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Dynamic multi-objective job shop scheduling: A genetic programming approach. In: A.S. Uyar, E. Ozcan, N. Urquhart (eds.) Automated Scheduling and Planning, *Studies in Computational Intelligence*, vol. 505, pp. 251–282. Springer (2013)
54. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Learning iterative dispatching rules for job shop scheduling with genetic programming. International Journal of Advanced Manufacturing Technology **67**(1-4), 85–100 (2013). DOI 10.1007/s00170-013-4756-9

55. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Transactions on Evolutionary Computation **18**(2), 193–208 (2014). DOI 10.1109/TEVC.2013.2248159
56. Nguyen, S., Zhang, M., Tan, K.C.: A dispatching rule based genetic algorithm for order acceptance and scheduling. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15, pp. 433–440. ACM, New York, NY, USA (2015). DOI 10.1145/2739480.2754821. URL http://doi.acm.org/10.1145/2739480.2754821
57. Nguyen, S., Zhang, M., Tan, K.C.: Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. 2015 IEEE Congress on Evolutionary Computation (CEC) (MAY), 2781–2788 (2015). DOI 10.1109/CEC.2015.7257234
58. Nie, L., Gao, L., Li, P., Zhang, L.: Application of gene expression programming on dynamic job shop scheduling problem. Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD) pp. 291–295 (2011). DOI 10.1109/CSCWD.2011.5960088
59. Nie, L., Shao, X., Gao, L., Li, W.: Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. International Journal of Advanced Manufacturing Technology **50**(5-8), 729–747 (2010). DOI 10.1007/s00170-010-2518-5
60. Park, J., Nguyen, S., Zhang, M., Johnston, M.: Genetic programming for order acceptance and scheduling. 2013 IEEE Congress on Evolutionary Computation, CEC 2013 (3), 1005–1012 (2013). DOI 10.1109/CEC.2013.6557677
61. Petrovic, S., Castro, E.: A Genetic Algorithm for Radiotherapy Pre-treatment Scheduling. Applications of Evolutionary Computation **6025**(August 2015), 462–471 (2010). DOI 10.1007/978-3-642-12242-2
62. Pfund, M., Fowler, J.W., Gupta, J.N.D.: a Survey of Algorithms for Single and Multi-Objective Unrelated Parallel-Machine Deterministic Scheduling Problems. Journal of the Chinese Institute of Industrial Engineers **21**(3), 230–241 (2004). DOI 10.1080/10170660409509404
63. Pickardt, C.W., Hildebrandt, T., Branke, J., Heger, J., Scholz-Reiter, B.: Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. International Journal of Production Economics **145**(1), 67–77 (2013). DOI 10.1016/j.ijpe.2012.10.016
64. Pinedo, M.: Scheduling Therory, Algorithms and Systems
65. Poli, R., Langdon, W.B., McPhee, N.F.: 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)
66. Seada, H., Deb, K.: U-NSGA-III : A Unified Evolutionary Algorithm for Single , Multiple , and Many-Objective Optimization pp. 1–30 (2014)
67. Tay, J.C., Ho, N.B.: Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Computers and Industrial Engineering **54**(3), 453–473 (2008). DOI 10.1016/j.cie.2007.08.008
68. Đurasević, M., Jakobović, D.: Comparison of solution representations for scheduling in the unrelated machines environment
69. Wang, Z., Tang, K., Yao, X.: Multi-objective approaches to optimal testing resource allocation in modular software systems. IEEE Transactions on Reliability **59**(3), 563–575 (2010). DOI 10.1109/TR.2010.2057310
70. Yu, L., Shih, H.M., Pfund, M., Carlyle, W.M., Fowler, J.W.: Scheduling of unrelated parallel machines: an application to pwb manufacturing. IIE Transactions **34**(11), 921–931 (2002). DOI 10.1023/A:1016185412209. URL http://dx.doi.org/10.1023/A:1016185412209
71. Zhang, M., Smart, W.: Learning weights in genetic programs using gradient descent for object recognition. In: Proceedings of the 3rd European Conference on Applications of Evolutionary Computing, EC'05, pp. 417–427. Springer-Verlag, Berlin, Heidelberg (2005)
72. Zhang, M., Wong, P.: Genetic programming for medical classification: a program simplification approach. Genetic Programming and Evolvable Machines **9**(3), 229–255 (2008). DOI 10.1007/s10710-008-9059-9. URL http://dx.doi.org/10.1007/s10710-008-9059-9
73. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. Evolutionary Computation, IEEE Transactions on **11**(6), 712–731 (2007). DOI 10.1109/TEVC.2007.892759
74. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation **1**(1), 32–49 (2011)

75. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm pp. 95–100 (2001). DOI 10.1.1.28.7571
76. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE T Evolut Comput **3**(4), 257–271 (1999). DOI 10.1109/4235.797969