Swarm and Evolutionary Computation 65 (2021) 100933

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Automatic preference based multi-objective evolutionary algorithm on vehicle fleet maintenance scheduling optimization

Yali Wang [a], Steffen Limmer [b], Markus Olhofer [b], Michael Emmerich [a,*], Thomas Bäck [a]

[a] Leiden Institute of Advanced Computer Science, Leiden University, Leiden 2333 CA, the Netherlands
[b] Honda Research Institute Europe GmbH, Offenbach 63073, Germany

ARTICLE INFO

Keywords:
DI-MOEA
Evolutionary algorithm
Multi-objective optimization
Preference
Scheduling optimization

ABSTRACT

A preference based multi-objective evolutionary algorithm is proposed for generating solutions in an automatically detected knee point region. It is named Automatic Preference based DI-MOEA (AP-DI-MOEA) where DI-MOEA stands for Diversity-Indicator based Multi-Objective Evolutionary Algorithm). AP-DI-MOEA has two main characteristics: firstly, it generates the preference region automatically during the optimization; secondly, it concentrates the solution set in this preference region. Moreover, the real-world vehicle fleet maintenance scheduling optimization (VFMSO) problem is formulated, and a customized multi-objective evolutionary algorithm (MOEA) is proposed to optimize maintenance schedules of vehicle fleets based on the predicted failure distribution of the components of cars. Furthermore, the customized MOEA for VFMSO is combined with AP-DI-MOEA to find maintenance schedules in the automatically generated preference region. Experimental results on multi-objective benchmark problems and our three-objective real-world application problems show that the newly proposed algorithm can generate the preference region accurately and that it can obtain better solutions in the preference region. Especially, in many cases, under the same budget, the Pareto optimal solutions obtained by AP-DI-MOEA dominate solutions obtained by MOEAs that pursue the entire Pareto front.

1. Introduction

The fleet maintenance scheduling optimization (VFMSO) problem was initially proposed in Wang et al. [1] due to the increasing demand by companies, corporations, and organizations of all sorts which rely on vehicle fleets to deliver products and services and need to maintain vehicles for safety reasons. In the problem, a vehicle fleet, such as a taxi fleet, a bus fleet, etc., can be maintained in multiple separate workshops according to a maintenance schedule. To be specific, each workshop has its own capacity and ability, meaning that on the one hand, each workshop has its own team and each team can work on only one car at the same time; on the other hand, each workshop is limited to the maintenance of the specific component(s) due to restrictions in the equipment or skill level of the staff. The maintenance schedule is optimized for each component based on its remaining useful lifetime (RUL) which has been predicted through predictive approaches or models [2]. Furthermore, the cost and time which are needed by different workshops are considered because it is possible that the maintenance of the same component produces different costs and workloads when the operation is performed in different workshops. The VFMSO problem is essential because it can not only ensure the safety of vehicles for use; at the same time, it can lead to low maintenance costs and longer lives for vehicles as well.

To enhance the approach in Wang et al. [1], to be specific, to handle the uncertainty in the problem and apply it on new application scenarios, in this paper, we improve it from the following two aspects:

1. There exists a lot of uncertainty when we use the predicted RUL for each component as its due date, because no matter how accurate the predictive model is, it is still possible that the component will break on other dates: before the due date or later. Therefore, instead of only the RUL, we decide to involve the RUL probability distribution as the foundation to assign the maintenance time in the scheduling optimization.
2. The VFMSO problem usually leads to a large and complex solution space, however, finding the most preferred solution is the ultimate goal. To this end, AP-DI-MOEA (Automatic Preference based DI-MOEA) is developed based on the framework of DI-MOEA (Diversity-Indicator based Multi-Objective Evolutionary Algorithm) [3]. The new algorithm can generate the preference region automatically and find solutions with a more fine-grained resolution in the preference region.

* Corresponding author.
E-mail addresses: y.wang@liacs.leidenuniv.nl (Y. Wang), steffen.limmer@honda-ri.de (S. Limmer), Markus.Olhofer@honda-ri.de (M. Olhofer), m.t.m.emmerich@liacs.leidenuniv.nl (M. Emmerich), t.h.w.Baeck@liacs.leidenuniv.nl (T. Bäck).

https://doi.org/10.1016/j.swevo.2021.100933
Received 4 May 2020; Received in revised form 23 March 2021; Accepted 10 June 2021
Available online 19 June 2021

This paper is organized as follows. Section 2 formulates the enhanced VFMSO problem. A literature review on preference based optimization is provided in Section 3. The customized multi-objective evolutionary algorithm for the enhanced VFMSO is introduced in Section 4, and in Section 5, we explain AP-DI-MOEA. Section 6 presents and discusses experiments and their results. Lastly, Section 7 concludes the paper and outlines directions for future work.

2. Problem formulation

For a car fleet operated by an operator, the components of cars (e.g., springs, brakes, tires or the engine) can fail and should be maintained regularly. Some separate workshops are available for the maintenance of the car fleet, and the repair time and maintenance cost are known for each component in each workshop. Beside the time and cost for repairing the car component, a fixed set-up cost and set-up time are also considered for each visit of a car to a workshop, which correspond to the cost and time required for the preparation of the maintenance operation.

The enhanced VFMSO problem addressed in this paper is defined as follows:

1. There are n cars $C = \{C_1, C_2, \dots, C_n\}$ and m workshops $W = \{W_1, W_2, \dots, W_m\}$.
2. Each car C_i comprises l_i components to be maintained for $i = 1, \dots, n$.
3. For each component O_{ij} ($j = 1, \dots, l_i$), i.e., the jth component of car C_i, there is a set of workshops capable of repairing it. The set of workshops is represented by W_{ij} which is a subset of W.
4. The processing time for maintaining component O_{ij} in workshop W_k is predefined and denoted by p_{ijk}.
5. The cost for maintaining component O_{ij} in workshop W_k is predefined and denoted by q_{ijk}.
6. The set-up time of car C_i in workshop W_k is predefined and denoted by x_{ik}.
7. The set-up cost of car C_i in workshop W_k is predefined and denoted by y_{ik}.
8. The number of teams in workshop W_k is predefined and denoted by z_k.
9. The previous repair time of component O_{ij} is recorded and denoted by L_{ij}.

At the same time, the following assumptions are made:

1. All workshops and teams are available at the time of the optimization and assumed to be continuously available.
2. All the components are independent from each other.
3. Times required for transport of cars from/to workshops are included in the maintenance time and cost of cars, and the set-up time.
4. Environmental changes (such as car accidents) are not considered here.
5. There are no precedence constraints among the components of different cars. Cars are maintained on a first-come-first-served basis.
6. Each team can only work on one operation at a time and an operation, once started, must run to completion.
7. No operation can start before the completion of the previous operation.

Two constraints are considered in the problem. As mentioned earlier, each workshop can only repair specific components, and this is the first constraint. Another constraint is that the maintenance periods of different operations for the same car should not overlap. It is obviously wrong if two overlapping maintenance operations of a car are assigned to different workshops because one car cannot be in two different workshops at the same time. If two overlapping maintenance operations of a car are assigned to the same workshop, it is not correct either because these two maintenance operations should be grouped together as one operation in this case. The grouping strategy will be explained in Section 4.

Three objectives are assumed to be relevant for the vehicle fleet operator, which are the total workload, total cost and expected number of failures. In a multi-objective optimization problem, the objectives typically are conflicting, i.e., achieving the optimal value for one objective requires some compromise on other objectives. In our problem, the fact that a faster maintenance usually is more expensive leads to the conflict between the first two objectives. The expected number of failures counts the times when the vehicles are broken on the road. Here, the expected value is used because the actual value is unknown at the time of the optimization due to uncertainties in the predictions. When the expected number of failures is large, less maintenance tasks are performed, therefore, the workload and cost can drop.

Let T_k denote the sum of the times spent for all operations that are processed in workshop W_k; M_i the sum of all costs spent for all maintenance operations of car C_i; F_{ij} the number of failures of component O_{ij}. Three objectives can be defined as:

$$\text{Minimize the total workload: } f_1 = \sum_{k=1}^{m} T_k \tag{1}$$

$$\text{Minimize the total cost: } f_2 = \sum_{i=1}^{n} M_i \tag{2}$$

Minimize the expected number of failures:

$$f_3 = \sum_{i=1}^{n} \sum_{j=1}^{l_i} \mathbb{E}(F_{ij}) \tag{3}$$

3. Literature review

Multi-objective scheduling optimization is a major topic in the research of manufacturing systems. Its fundamental task is to organize work and workloads to achieve comprehensive optimization in multiple aspects, such as the processing time, processing cost and production safety, by deploying resources, setting maintenance time and processing sequence. In past decades, this issue has received a great deal of interest and research in different fields, such as scheduling of charging/discharging for electric vehicles [4]; scheduling in cloud computing [5]; scheduling of crude oil operations [6]; scheduling in the manufacturing industry to reduce carbon emissions [7]; scheduling medical treatments for resident patients in a hospital [8]; scheduling for Internet service providers [9], and so on.

As a typical workshop style, the flexible job shop scheduling problem (FJSP) is an essential branch of production planning problems. The FJSP consists of a set of independent jobs to be processed on multiple machines, and each job contains several operations with a predetermined order. It is assumed that each operation must be processed in specified processing time on a specific machine out of multiple alternatives. The problem has been extensively studied in the literature (for example, Chiang and Lin [10], Yuan and Xu [11], Gao et al. [12]). The FJSP is the research basis of the maintenance scheduling optimization problem and many real-world problems extend the standard FJSP by adding specific features. Özgüven et al. [13] considers FJSP-PPF (process plan flexibility), where jobs can have alternative process plans. It is assumed that the process plans are known in advance and that they are represented by linear precedence relationships. Because only one of the alternative plans has to be adopted for each job, the FJSP-PPF deals with not only routing and sequencing sub-problems, but also the process plan selection sub-problem. In this paper, a mixed-integer linear programming model is developed for the FJSP-PPF. In [14], a mathematical model and a genetic algorithm are proposed to handle the feature of overlapping in operations. It is assumed that a lot which contains a batch of identical items is transferred from one machine to the next only when all items in the lot have completed their processing, therefore, sublots are transferred from one machine to the next for processing without waiting for the entire lot to be processed at the predecessor machine, meaning that starting a successor operation of job is not necessary to finish of its predecessor completely. Three features are considered in Yu et al. [15],

which are (1) job priority; (2) parallel operations: some operations can be processed simultaneously; (3) sequence flexibility: the sequence of some operations can be exchanged. A mixed integer linear programming formulation (MILP) model is established to formulate the problem and an improved differential evolution algorithm is designed. Because of unexpected events occurring in most of the real manufacturing systems, there is a new type of scheduling problem known as the dynamic scheduling problem. This type of problem considers random machine breakdowns, adding new machines, new job arrival, job cancellation, changing processing time, rush order, rework or quality problem, due date changing, etc. Corresponding works on the FJSP include [16–19]. Compared with the standard FJSP, our VFMSO problem has some special properties: (1) flexible sequence: the sequence of the components is not predefined, but mainly influenced by the RUL probability distribution. (2) multiple problem parameters: besides the processing time, other problem parameters like the maintenance cost, set-up time, set-up cost, repair teams, etc, also have impacts on the result.

Our real-world problem, like many other multi-objective optimization problems, can lead to a large objective space. However, finding a well-distributed set of solutions on the Pareto front requests a large population size and computational effort. Therefore, instead of spreading a limited size of individuals across the entire Pareto front, we decide to only focus on a part of the Pareto front, to be specific, the search for solutions will be only guided towards the preference region which, in our algorithm, is determined by the knee point. It has been argued in the literature that knee points are most interesting solutions, naturally preferred solutions and most likely the optimal choice of the decision maker (DM) [20–23].

The knee point is a point for which a small improvement in any objective would lead to a large deterioration in at least one other objective. In the last decade, several methods have been presented to identify knee points or knee regions. Das [20] refers the point where the Pareto surface “bulges” the most as the knee point, and this point corresponds to the farthest solution from the convex hull of individual minima which is the minima of the single objective functions. Zitzler [24] defines ϵ-dominance: a solution a is said to ϵ-dominate a solution b if and only if $f_i(a) + \epsilon \geq f_i(b)\ \forall i = 1, \ldots, m$ where m is the number of objectives. A solution with a higher ϵ-dominance value with respect to the other solutions in the Pareto front approximation, is a solution having higher trade-offs and in this definition corresponds to a knee point. The authors of [25] propose to calculate the density of solutions projected onto the hyperplane constructed by the extreme points of the non-dominated solutions, then identify the knee regions based on the solution density.

Different algorithms of applying knee points in MOEA have also been proposed. Branke [23] modifies the second criterion in NSGA-II [26], and replaces the crowding distance by either an angle-based measure or a utility-based measure. The angle-based method calculates the angle between an individual and its two neighbors in the objective space. The smaller the angle, the more clearly the individual can be classified as a knee point. However, this method can only be used for two objective problems. In the utility-based method, a marginal utility function is suggested to approximate the angle-based measure in the case of more than two objectives. The larger the external angle between a solution and its neighbors, the larger the gain in terms of linear utility obtained from substituting the neighbors with the solution of interest. However, the utility-based measure is not suited for finding knees in concave regions of the Pareto front.

Rachmawati [27,28] proposes a knee-based MOEA which computes a transformation of original objective values based on a weighted sum niching approach. The extent and the density of coverage of the knee regions are controllable by the parameters for the niche strength and pool size. The strategy is susceptible to the loss of less pronounced knee regions.

Schütze [29] investigates two strategies for the approximation of knees of bi-objective optimization problems with stochastic search algorithms. Several new definitions for identifying knee points and knee regions for bi-objective optimization problems has been suggested in Deb and Gupta [30] and the possibility of applying them has also been discussed.

Besides the knee points, the reference points, which are normally provided by the DM, have also been used to find a set of solutions near reference points. Deb [31] proposes an MOEA, called R-NSGA-II, by which a set of Pareto optimal solutions near a supplied set of reference points can be found. The dominance relation together with a modified crowding distance operator is used in this methodology. For all solutions of the population, the distances to all reference points are calculated and ranked. The lowest rank (over all reference points) of a solution is used as its crowding distance. Besides, a parameter ϵ is used to control the spread of obtained solutions. Recently, R-NSGA-II was extended and the reference point based NSGA-III (R-NSGA-III) is proposed for solving higher objective problems [32]. Bechikh proposes KR-NSGA-II [33] by extending R-NSGA-II. Instead of obtaining the reference points from the DM, in KR-NSGA-II, the knee points are used as mobile reference points and the search of the algorithm was guided towards these points. The number of knee points of the optimization problem is needed as prior information in KR-NSGA-II.

Gaudrie [34] uses the projection (intersection in case of a continuous front) of the closest non-dominated point on the line connecting the estimated ideal and nadir points as default preference. Conditional Gaussian process simulations are performed to create possible Pareto fronts, each of which defines a sample for the ideal and the nadir point, and the estimated ideal and nadir are the medians of the samples.

Rachmawati and Srinivasan [35] evaluate the worthiness of each non-dominated solution in terms of compromise between the objectives. The local maxima is then identified as potential knee solutions and the linear weighted-sums of the original objective functions are optimized to guide solutions toward the knee regions.

Another idea of incorporating preference information into evolutionary multi-objective optimization is proposed in Thiele et al. [36]. They combine the fitness function and an achievement scalarizing function containing the reference point. In this approach, the preference information is given in the form of a reference point and an indicator-based evolutionary algorithm IBEA [37] is modified by embedding the preference information into the indicator. Various further preference based MOEAs have been suggested, e.g., [38–40].

In our proposed algorithm, i.e., AP-DI-MOEA, we adopt the method from [20] to identify the knee point, design the preference region based on the knee point, and guide the search towards the preference region. The advantages of our algorithm are: (1) no prior knowledge is used in identifying the knee point and knee region; (2) the preference region is generated automatically and narrowed down step by step to benefit its accuracy; (3) our strategy cannot only handle bi-objective optimization problems, but also tri- and many-objective problems; (4) although we integrate the strategy with DI-MOEA, it may be integrated with any standard MOEAs (such as NSGA-II [26], SMS-EMOA [41] and others); (5) the proposed algorithm is capable of finding preferred solutions for multi-objective optimization problems with linear, convex, concave Pareto fronts and discrete problems.

4. Customized algorithm for vehicle fleet maintenance scheduling optimization

For our real-world VFMSO problem, we first define the execution window for each component based on its predicted RUL probability distribution which is assumed to be a normal distribution. The execution window suggests that the maintenance of the component can only start at a time spot inside the window. The mean (μ) and standard deviation (σ) of the predicted RUL probability distribution determine the interval of the execution window, which is defined as: $[\mu - 2 \times \sigma, \mu + 2 \times \sigma]$. The interval is chosen relatively long because 95% of the values are within two standard deviations of the mean, therefore, maintenance before or after the interval hardly makes sense.

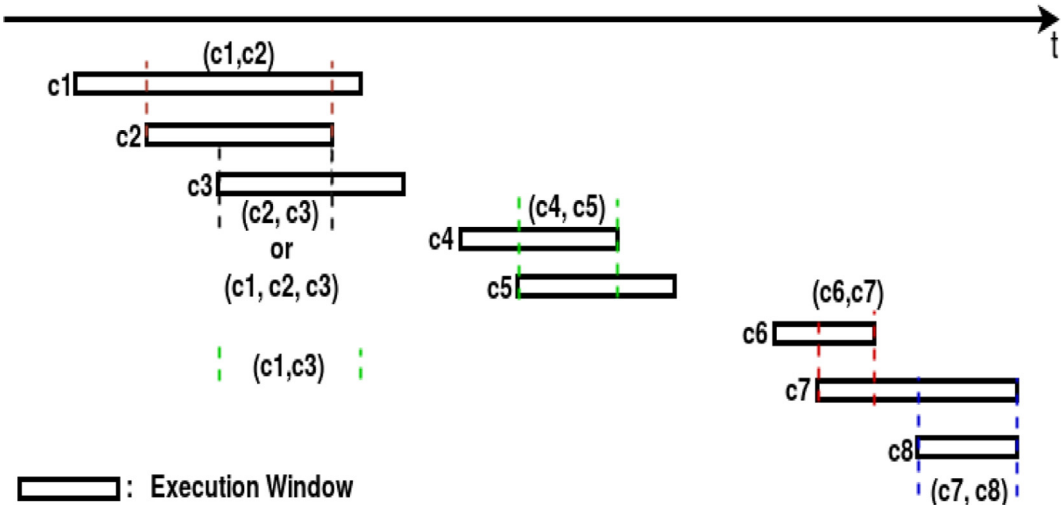


Fig. 1. Possible groups for a car with eight components.

After the determination of the execution window, the following two special strategies have been taken to improve the process of scheduling optimization:

- Grouping components.
- Obtaining the penalty cost and expected number of failures by Monte Carlo simulation.

Lastly, the evolutionary algorithm (EA) is chosen to solve this real-world application problem due to its powerful characteristics of robustness and flexibility to capture global solutions of complex combinatorial optimization problems. Moreover, EAs are well suited to solve multi-objective optimization problems due to their ability to approximate the entire Pareto front in a single run.

4.1. Grouping components

It would be troublesome and also a waste of time and effort to send a car to workshops repeatedly in a short period of time to repair different components. In our algorithm, since each component has its execution window for its maintenance, it is possible to combine the maintenance of several components to one visit if their execution windows overlap. Especially, by grouping the maintenance of multiple components into one maintenance operation, the set-up cost and set-up time are charged only once for the complete group of components.

Fig. 1 represents the execution windows of eight components of a car. The overlap of the execution windows shows the possibility of grouping these components. Combining components can only be effective if there is a common overlap of the execution window for components from the same car, and the starting time of the group operation must lie within the common overlap. In this example, component $c1$ can be grouped with $c2$ and/or $c3$ due to the overlap between their execution windows. Other possible group structures can be deduced in the same manner.

4.2. Monte Carlo simulation

Within the execution window of a component, an arbitrary time can be chosen as the starting time for maintaining the component. The maintenance time of each component should be as close as possible to its real due date, because:

- Performing the maintenance too early results in higher maintenance costs in the long term, because more maintenance tasks have to be done.
- The risk of breaking down on the road will increase if the maintenance date is too late.

Therefore, we use Monte Carlo simulation to simulate the "real" due dates for each component. In our experiments, stability can be achieved at a few hundred samples, in our case, 1000 samples of the due date are generated in the execution window of each component according to its predicted RUL probability distribution (see Section IV). Fig. 2 shows an example of the execution window evolved from the predicted RUL probability distribution of a component. After 1000 sampled due dates are generated in the execution window, the scheduled maintenance date of the component is compared with these samples one by one, and each comparison can lead to three situations. Let us use d_{ij}^{v} to denote the vth due date sample of component O_{ij}; and D_{ij} the scheduled maintenance date of component O_{ij}. Three possibilities after the comparison are:

Case 1) $D_{ij} < d_{ij}^{v}$

The scheduled maintenance date is earlier than the sample (or the "real" due date) means that the component will be maintained before it is broken. In this case, its useful life between the maintenance date and the due date will be wasted. Therefore, a corresponding penalty cost is imposed to reflect the waste. To calculate the penalty cost, a linear penalty function is suggested based on the following assumptions:

- If a component is maintained when it is new or the previous maintenance has just completed, the penalty cost would be the full cost of maintaining it, which is $c + s$: the maintenance cost of the component and the set-up cost of the car;
- If a component is maintained at exactly its due date, the penalty cost would be 0.

Assume d_{ij}^{v} is "Sampled Due date" in Fig. 2, and D_{ij} is "Maintenance date a", in this case, D_{ij} is earlier than d_{ij}^{v}. The penalty cost of "Maintenance date a" for "Sampled Due date" would be the vertical dotted line above "Maintenance date a".

Case 2) $D_{ij} > d_{ij}^{v}$

The scheduled maintenance date is later than the sample means that the maintenance date is too late and the defect occurs on the use. Still, d_{ij}^{v} is "Sampled Due date" in Fig. 2, but the scheduled maintenance date D_{ij} is "Maintenance date b". In this case, D_{ij} is later than d_{ij}^{v}, and the vehicle will break down on the road. In our algorithm, the number of failures will be increased by one.

Case 3) $D_{ij} = d_{ij}^{v}$

The ideal situation is that the maintenance date is scheduled on the due date. The component can be maintained exactly at the date that the component is broken. In this case, there is no penalty or failure.

The averages of the penalty costs and the number of failures from 1000 due date samples will be used as the penalty cost and expected number of failures for the scheduled maintenance date of the compo-

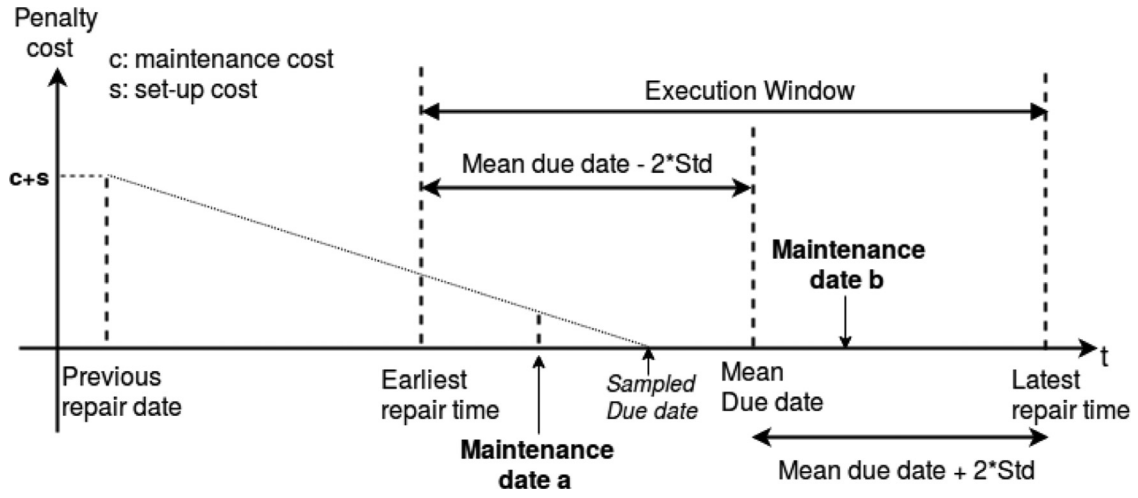


Fig. 2. Execution window of a component.

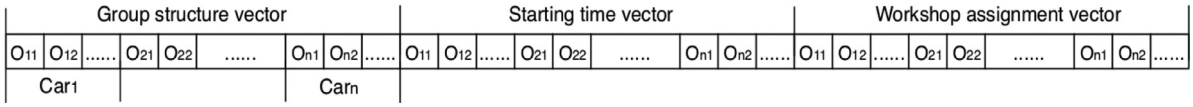


Fig. 3. Three-vector chromosome.

nent. For each operation (the single-component operation or group operation), its cost consists of three parts: the set-up cost of the car, the maintenance costs and the penalty costs of all components of the operation. The penalty cost of components is a part of the total cost, and the expected number of failures of components is the third objective to be minimized in our multi-objective optimization.

4.3. Implementation of evolutionary algorithm operators

To solve our application problem with an EA, there are several basic issues we need to deal with, such as, how to represent an individual or solution in the population (Chromosome Encoding); how to take these chromosomes into a process of evolution (Genotype-Phenotype Mapping); how to create variations of solutions in each iteration (Genetic Operators). Details of these topics are given in the following subsections.

4.3.1. Chromosome encoding

In our algorithm, a three-vector chromosome (Fig. 3) is proposed to represent an individual, and the three vectors are:

- Group structure vector: the group structures of components.
- Starting time vector: the starting times of operations.
- Workshop assignment vector: the workshops for operations.

The group structure vector gives the information of which components are in the same group, it is initialized by randomly picking a feasible group structure for each car (check the details in Wang et al. [1]). The generation of the starting time vector should be later than the generation of the group structure vector because the starting time of each operation is determined by the execution window which is the entire execution window of the component for a single-component operation or the execution window intersection for a group operation. A time spot is randomly selected from the execution window or execution window intersection for each operation in order to initialize the starting time vector.

A workshop is considered as “several workshops” based on its capacity (the number of teams). By this way, the schedule for each workshop team can be achieved from the solution. For example, consider that two workshops have three and four repairing teams respectively. Then, group operations can be randomly assigned to seven “workshops”, the former three and the latter four represent the corresponding teams in two workshops.

4.3.2. Genotype-phenotype mapping

To use the power of EAs to obtain a better population, we need to evaluate each chromosome and give the better ones higher probabilities to produce offspring. This is done by genotype-phenotype mapping or decoding the chromosome. In our problem, it is to convert an individual into a feasible schedule to calculate the objectives and constraints which represent the relative superiority of a chromosome. The genotype-phenotype mapping can be easily achieved in our algorithm because the group structure, the starting time and the workshop team of the operations can be acquired directly from each individual. When converting an individual into a schedule, it is possible that the processing times of two or more operations assigned to the same workshop team are overlapping since the starting time of each operation is decided in the starting time vector. In this situation, the principle of first-come-first-served is followed: the starting time and processing time of the earlier started operation remain the same; the starting time of the later started operation is delayed until the completion of the previous operation; the processing time of the later started operation remains the same; while, an extra waiting time is added to the later started operation as a penalty because the vehicle waits in the workshop for the maintenance.

4.3.3. Genetic operators

In accordance with the problem and its encoding, specific crossover and mutation operators have been designed for our problem (check the details in Wang et al. [1]). Both operators are applied separately to the three parts of the chromosome.

For the group structure vector, multi-point crossover can be used as crossover operator and the number of cutting points depends on the length of the vector. The same cutting points can be applied to the starting time vector when performing crossover. However, the change on the group structure vector as a consequence of the crossover may result in the invalidity of genes in the starting time vector because it is possible that the group members and execution window intersections have changed due to the new group structure. Therefore, when performing

the crossover on the starting time vector, the starting times of all operations should be checked based on the new group structure and a new starting time is produced randomly from the correct intersection in the case that the starting time of an operation is invalid. The multi-point crossover can be applied to the workshop assignment vector as well.

The mutation operator alters one or more gene values in a chromosome. Similarly, the mutation should be operated on the group structure vector first due to its impact on the starting time vector; the starting time of operations should be checked and corrected after the mutation is done on the group structure vector. Afterwards, several gene values can be altered in the starting time vector and workshop assignment vector to generate a new individual.

5. Proposed preference based algorithm

As the number of objectives and decision variables increases, the number of non-dominated solutions tends to grow exponentially [42]. This brings more challenges on achieving efficiently a solution set with satisfactory convergence and diversity. At the same time, a huge number of solutions is needed to approximate the entire Pareto front. However, a big population means more computational time and resources. To overcome these difficulties, we propose an automatic preference based MOEA, which can generate the preference region or the region of interest (ROI) automatically and find non-dominated solutions in the preference region instead of the entire Pareto front. The automatic preference based MOEA is developed based on the framework of DI-MOEA (Diversity-Indicator based Multi-Objective Evolutionary Algorithm) [3]. We call our new algorithm AP-DI-MOEA.

DI-MOEA is an indicator-based MOEA, it has shown to be competitive to other MOEAs on common multi-objective benchmark problems. Moreover, it is invariant to the shape of the Pareto front and can achieve evenly spread Pareto front approximations. DI-MOEA adopts a hybrid selection scheme:

- The $(\mu + \mu)$ generational selection operator is used when the parent population can be layered into multiple dominance ranks. The intention is to accelerate convergence until all solutions are non-dominated.
- The $(\mu + 1)$ steady state selection operator is adopted in the case that all solutions in the parent population are mutually non-dominated and the diversity is the main selection criterion to achieve a uniform distribution of the solutions on the Pareto front.

DI-MOEA employs non-dominated sorting as the first ranking criterion; the diversity indicator, i.e., the Euclidean distance based geometric mean gap indicator, as the second, diversity-based ranking criterion to guide the search. Two variants of DI-MOEA, denoted as DI-1 and DI-2, exist, which use the crowding distance and diversity indicator, respectively, as the second criteria in the $(\mu + \mu)$ generational selection operator. While, to ensure the uniformity of the final solution set, the diversity indicator is used by both variants in the $(\mu + 1)$ steady state selection operator. Analogously, two variants of AP-DI-MOEA, i.e., AP-DI-1 and AP-DI-2, are derived from the two variants of DI-MOEA.

The workings of AP-DI-MOEA are outlined in Algorithm 1. In the algorithm, the variable $evals\_update$ is used to record the condition of generating or updating the preference region. Its initial value is assigned to $divide\_size$ (line 4 in Algorithm 1). Exceedance of $divide\_size$ is a predefined condition to divide the algorithm into two phases: learning phase and decision phase. In the learning phase, the algorithm explores the possible area of Pareto optimal solutions and finds the rough approximations of the Pareto front. In the decision phase, the algorithm identifies the preference region and finds preferred solutions. When the algorithm starts running and the number of evaluations reaches or exceeds $divide\_size$ at some moment, the first preference region will be generated and $evals\_update$ will be updated for determining a new future moment when the preference region needs to be updated (line 12–16 in Algorithm 1). The process of updating $evals\_update$ repeats until the end to narrow down the preference region step by step. The first value of $evals\_update$, i.e., $divide\_size$, is a boundary line. Before it is satisfied, AP-DI-MOEA runs exactly like DI-MOEA to approximate the whole Pareto front; while, after it is satisfied, the preference region is generated automatically and AP-DI-MOEA finds solutions focusing on the preference region. The subsequent values of $evals\_update$ define the later moments to update the preference region; eventually, a precise ROI with a proper size can be achieved.

The first/new preference region is formed based on the population at the moment when the condition of $evals\_update$ is satisfied. To be specific, the preference region is determined by the knee point of the current population. Algorithm 2 gives the details of line 14 in Algorithm 1, it introduces the steps of finding the knee point of a non-dominated solution set and constituting a hypercube shaped preference region according to the knee point. Fig. 4 also gives an illustration of finding the knee point in bi-dimensional space. Firstly, the upper quartile objective values (line 12 in Algorithm 2) in the solution set are used as a boundary to define outliers. To identify the knee point, solutions outside this boundary are removed from the solution set (line 15–19 in Algorithm 2). The extreme solutions (the solutions with the maximum value in one objective) are then found inside the boundary (line 22 in Algorithm 2) and a hyperplane is formed based on the extreme solutions (line 23 in Algorithm 2). In a bi-dimensional space (Fig. 4), the hyperplane is only a line connecting two extreme solutions. According to the numbers of points below and above the hyperplane (line 24 and 25 in Algorithm 2), the shape of the solution set can be roughly perceived. We will distinguish between “convex” and “concave” regions. Points in the *convex* (*concave*) *region* are dominating (dominated by) at least one point in the hyperplane spanned by the extreme points. However, when the number of the points in the convex region and the number of points in the concave region is close enough, it implies that the shape of the current solution set is almost linear. This occurs both when the true Pareto front is linear and when the solution set converges very well in a small area of the Pareto front. A parameter ϵ then is used to represent the closeness and it is a small number decided by the size of the solution set. In the case that the shape of the current solution set is (almost) linear, the solution with the largest hypervolume value with regards to the worst objective vector (i.e., $L[i]$ in line 13 in Algorithm 2) is adopted as the knee point (line 33–39 in Algorithm 2). While, under the condition that the shape of the current solution set is convex or concave, the knee point is identified by the method in Das [20]. The solution in the convex or concave region with the largest Euclidean distance to the hyperplane is chosen as the knee point (line 42–45 in Algorithm 2). After the knee point is found, the preference region can be determined based on the knee point by the following formula:

$$P\_region[i] = knee[i] + (L[i] - knee[i]) \times 85\% \tag{4}$$

Let i denotes the ith objective, as in Algorithm 2, $L[i]$ is the worst value of the ith objective in the population, $knee[i]$ is the ith objective value of the knee point and $P\_region[i]$ is the upper bound of the preference region. W.l.o.g. We assume the objectives are to be minimized and the lower bound of the preference region is the origin point. According to the formula, we can see that the first preference region is relatively large (roughly 85% of the entire Pareto front). With the increase in the number of iterations, the preference region will be updated and become smaller and smaller because every preference region picks 85% of the current Pareto front. Eventually, we want the preference region to reach a proper range, say, 15% of the initial Pareto front. The process of narrowing down the preference region step by step can benefit the accuracy of the preference region.

Algorithm 1 only shows the workings of AP-DI-2. The difference between AP-DI-1 and AP-DI-2 is the same as the difference between DI-1 and DI-2, i.e., the diversity criterion in the $(\mu + \mu)$ generational selection operator in AP-DI-1 is the crowding distance, in AP-DI-2 it is the diversity indicator. In the algorithm, the initialized population is sorted based on non-domination and the diversity value of each solution is cal-

```
   Inputs:
   popsize;      // population size
   divide_size;      // number of evaluations before generating 1st preference region
   batch_size;      // number of evaluations between preference region updates
 1: P_0 ← init(popsize);      // initialize random population
 2: existRegion ← false;      // indicates whether a preference region was already computed
 3: evals ← 0;      // number of evaluations to far
 4: evals_update ← divide_size;      // number of evaluations when 1st preference region is computed
 5: (R_1, ..., R_{ℓ_0}) ← non_dominated_sorting(P_0);      // partition into fronts of increasing dominance ranks
 6: for each i ∈ {1, ..., ℓ_0} do
 7:    calculate diversity indicator for all solutions on R_i;
 8: end for
 9: t ← 0;
10: while Stop criterion not satisfied() do
11:    // update / computation of preference region
12:    if (evals > evals_update && ℓ_t == 1) then
13:       existRegion ← true;
14:       calculate P_region;      //generate a (new) preference region, i.e., Algorithm 2
15:       evals_update ← evals_update + batch_size;
16:    end if
17:
18:    // offspring generation
19:    if (ℓ_t > 1 || t == 0) then
20:       //(μ + μ) generational scheme
21:       Q_t ← Gen(P_t);      // generate popsize offspring by recombination and mutation
22:       evaluate(Q_t);
23:       evals ← evals + popsize;
24:       M_{t+1} = P_t ∪ Q_t;      // combine offspring and parent population
25:    else
26:       // (μ + 1) steady state generational scheme
27:       q ← Gen(P_t);      // generate only one offspring by recombination and mutation
28:       evaluate(q_t);
29:       evals ← evals + 1;
30:       M_{t+1} = P_t ∪ {q_t};      // combine offspring and parent population
31:    end if
32:
33:    // construction of new population based on non-dominated sorting
34:    (R_1, ..., R_{ℓ_{t+1}}) ← non_dominated_sorting(M_{t+1});
35:    P_{t+1} ← ∅;
36:    i ← 0;
37:    while |P_{t+1}| < popsize do
38:       i ← i + 1;
39:       P_{t+1} ← P_{t+1} ∪ R_i;
40:    end while
41:
42:    // truncation of new population based on further ranking criterion(s)
43:    if (|P_{t+1}| > popsize) then
44:       if (existRegion == false) then
45:          rank solutions in R_i by diversity indicator contribution;
46:       else
47:          rank solutions in R_i by diversity indicator contribution and Euclidean distance to knee point;
48:          assign the lowest possible rank to all solutions in R_i, which are outside P_region;
49:       end if
50:       n ← |P_{t+1}| − popsize;
51:       remove the n solutions in R_i with lowest ranks from P_{t+1};
52:    end if
53:
54:    t ← t + 1;
55:    l_t ←the number of fronts of P_t;
56: end while
```

Algorithm 1. AP-DI-MOEA-2

culated to be in later parent selection (line 5–8 in Algorithm 1). When evolution of the population takes place, according to different phases of optimization, the (μ + μ) generational selection operator generates multiple offspring in one iteration to explore more decision space and push the population quickly towards the Pareto front (line 20–24 in Algorithm 1); the (μ + 1) steady state selection operator generates only one offspring in order to achieve a uniformly distributed set (line 26–30 in Algorithm 1). To achieve the next generation population, the combination of parents and offspring is classified into different layers according to the non-dominance relation. The points from the first non-domination front are preserved, continuing with points in the second non-domination front, until the number of points reaches the population size (line 34–40 in Algorithm 1). Under the case that the number of points surpasses the population size, a truncation selection is carried out (line 43–52 in Algorithm 1). When there is no preference region, the population will be truncated based on the diversity indicator. While, if a

Inputs:
$popsize$; // population size
n; // number of objectives
P_t; // current population
ϵ; // parameter (>0) for distinguishing convex/concave shape

declare($Q[n]$); //upper quartile objective values of P_t
declare($L[n]$); //worst objective values of P_t
declare($knee[n]$); //knee point of P_t
declare($P\_region[n]$); //preference region of P_t
declare($Epoints[n][n]$); //extreme points (single-objectives)
$foundknee \leftarrow false$; // indicates whether the knee point was already found
$P'_t \leftarrow P_t$; //copy the current population for finding the knee

// remove outliers with lowest 25% of objective values
for each $i \in \{1, \ldots, n\}$ **do**
  sort(P'_t) by the ith objective in ascending order;
  $Q[i] \leftarrow P'_t$.get_index($\frac{3}{4} \times popsize$).get_obj($i$); //upper quartile value of the ith objective
  $L[i] \leftarrow P'_t$.get_index($popsize$).get_obj(i); //the largest (worst) value of the ith objective
end for
for all solution $s \in P'_t$ **do**
  if s.get_obj($i = 1, ..., n$) $> Q[i]$ **then**
    remove s from P'_t;
  end if
end for

//find knee point by computing distance to hyperplane
$Epoints[\bullet][\bullet] \leftarrow$ extreme points in P'_t;
hyperplane($Epoints[\bullet][\bullet]$); //generate hyperplane by $Epoints[\bullet][\bullet]$
$num_a \leftarrow$ number of points in concave region of hyperplane;
$num_v \leftarrow |P'_t| - num_a$; // number of points in convex region
if $(num_v - num_a > \epsilon)$ **then**
  //roughly convex shape
  remove solutions in concave region from P'_t;
else if $(num_a - num_v > \epsilon)$ **then**
  //roughly concave shape
  remove solutions in convex region from P'_t;
else
  //roughly linear shape
  //find knee point by computing hypervolume
  for all solution $s \in P'_t$ **do**
    calculate hypervolume of s with reference point $L[\bullet]$;
  end for
  $knee[\bullet] \leftarrow$ solution with the largest hypervolume value;
  $foundknee \leftarrow true$;
end if
if $(foundknee == false)$ **then**
  for all solution $s \in P'_t$ **do**
    calculate distance between s and hyperplane;
  end for
  $knee[\bullet] \leftarrow$ solution with the largest distance;
end if

//determine current preference region by knee point.
for each $i \in \{1, \ldots, n\}$ **do**
  $P\_region[i] \leftarrow knee[i] + (L[i] - knee[i]) \times 85\%$
end for

Algorithm 2. Finding the knee point and defining the preference region.

preference region already exists, the population will be truncated based on first the diversity indicator, then Euclidean distance to the knee point. In this process, the diversity indicator contribution and Euclidean distance to the knee point are calculated only for the solutions in the preference region. Solutions outside of the region are given relatively larger values and they will be eliminated soon in the optimization process.

There are different strategies to update the values of $evals\_update$. In our algorithm, we divide the whole computing budget into two parts, the first half is used to find an initial entire Pareto front approximation, and the second half is used to update the preference region and find solutions in the preference region. Assume the total computing budget is $budget\_size$ (the number of evaluations), then the first value of $evals\_update$ is $\frac{1}{2} \times budget\_size$. Due to the reason that we expect a final preference region with a size of around 15% of the initial entire Pareto front and each new preference region takes 85% of the current Pareto front, according to the formula: $0.85^{12} \approx 0.14$, the value of $evals\_update$ can be updated by the following formula:

$$\begin{aligned} evals\_update &= evals\_update + batch\_size \\ &= evals\_update + (budget\_size/2)/12 \end{aligned} \tag{5}$$

Another half of the budget can be divided into 12 partial-budgets and a new preference region is constituted after each partial-budget. In

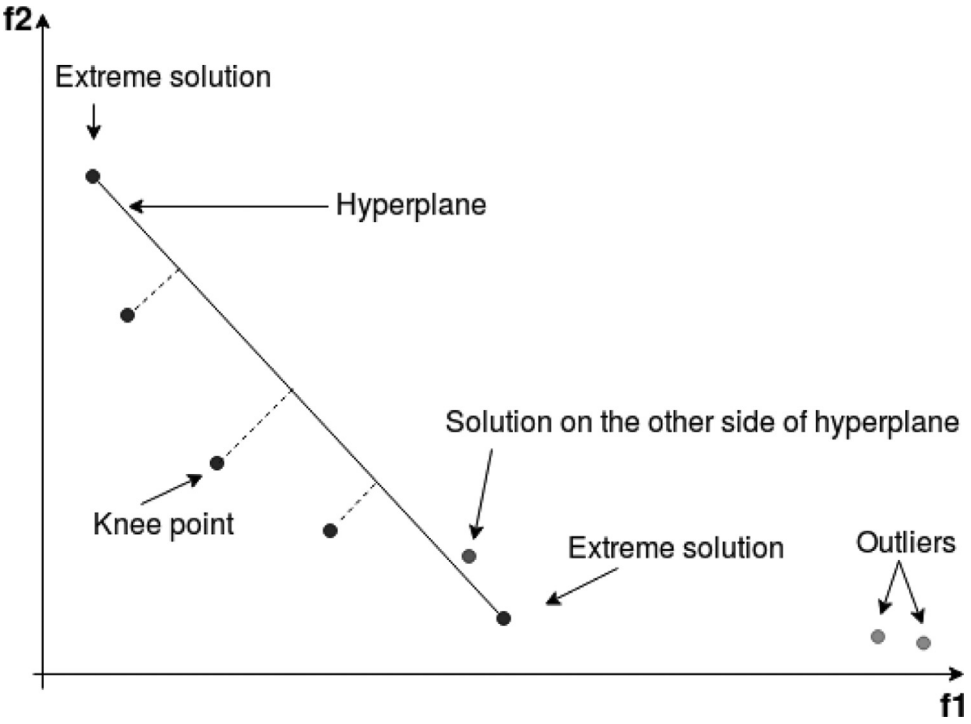


Fig. 4. Finding the knee point in bi-dimensional space.

the end, the final preference region is achieved and solutions focusing on this preference region are obtained.

6. Experimental results

6.1. Experimental design

In this section, simulations are conducted to demonstrate the performance of proposed algorithms on both benchmark problems and our real-world application problems. All experiments are implemented based on the MOEA Framework (http://www.moeaframework.org/), which is a Java-based framework for multi-objective optimization.

For the two variants of AP-DI-MOEA: AP-DI-1 and AP-DI-2, their performances have been compared with DI-MOEA: DI-1, DI-2 and NSGA-III [43]. We compare our algorithm with NSGA-III because NSGA-III is a representative state-of-the-art evolutionary multi-objective algorithm and it is very powerful to handle problems with non-linear characteristics. For bi-objective benchmark problems, algorithms are tested on ZDT1 and ZDT2 with 30 variables. For three objective benchmark problems, DTLZ1 with 7 variables and DTLZ2 with 12 variables are tested. These benchmark problems are chosen to verify the selection mechanism in our algorithm because they all resemble the shape (or reversal shape) of the Pareto front of the application problem. Besides, these continuous benchmarks are adopted because they are very familiar to the community of evolutionary multi-objective optimization (EMO). Ideally, the discrete nature of the problem would also be reflected in the benchmark, but the combinatorial problems found in the EMO community are not very general and resemble only very specific applications or are very simple toy problems, such as LOTZ. For the real-world application problem of VFMSO, experiments have been conducted on two instances with different sizes. The configurations of the two instances, such as the predicted RUL probability distribution, the processing time and maintenance cost of each component, the set-up time and cost of each car, are made available on http://moda.liacs.nl. On every problem, we run each algorithm 30 times with different seeds, while the same 30 different seeds are used for all algorithms. All the experiments are performed with a population size of 100; and for bi-objective problems, experiments are run with a budget of $22,000$ (objective function) evaluations, DTLZ three objective problems with a budget of $120,000$ evaluations, the VFMSO problems with a budget of $1,200,000$ evaluations. This setting is chosen to be more realistic in the light of the applications in scheduling that we ultimately want to solve.

6.2. Experiments on bi-objective problems

Bi-objective problems are optimized with a total budget of $22,000$ evaluations, when the number of evaluations reaches $10,000$ times, the first preference region is generated, then after every 1200 evaluations, the preference region will be updated. Fig. 5 shows the Pareto front approximations from a typical run on ZDT1 (left column) and ZDT2 (right column). The graphs on the upper row are obtained from DI-1 and AP-DI-1, while the graphs on the lower row are from DI-2 and AP-DI-2. In each graph, the entire Pareto front approximation from DI-MOEA and the preferred solutions from AP-DI-MOEA (or *AP solutions*) are presented, at the same time, the preference region of AP-DI-MOEA is also shown by the gray area.

Besides the visualization of the Pareto fronts, we also compute the knee point of the entire final Pareto front approximation from DI-MOEA via the strategy described in Algorithm 2. For each run of DI-MOEA and AP-DI-MOEA with the same seed, the following two issues have been checked:

- If the knee point from DI-MOEA is in the preference region achieved by its derived AP-DI-MOEA;
- If the knee point from DI-MOEA is dominated by or dominating AP solutions; or if it is a non-dominated solution (mutually non-dominated with all AP solutions).

Table 1 shows the results of 30 runs. For ZDT1 problem, all 30 knee points from DI-1 and DI-2 are in the preference regions from AP-DI-1 and AP-DI-2 respectively; in all these knee points, 10 from DI-1 and 7 from DI-2 are dominated by AP solutions. For ZDT2 problem, most knee points are not in corresponding preference regions, but for those in the preference regions, almost all of them are dominated by AP solutions. Please note that when a knee point from DI-MOEA is outside of the preference region from AP-DI-MOEA, it is not possible that it can dominate any AP solutions because all AP solutions are in the preference region

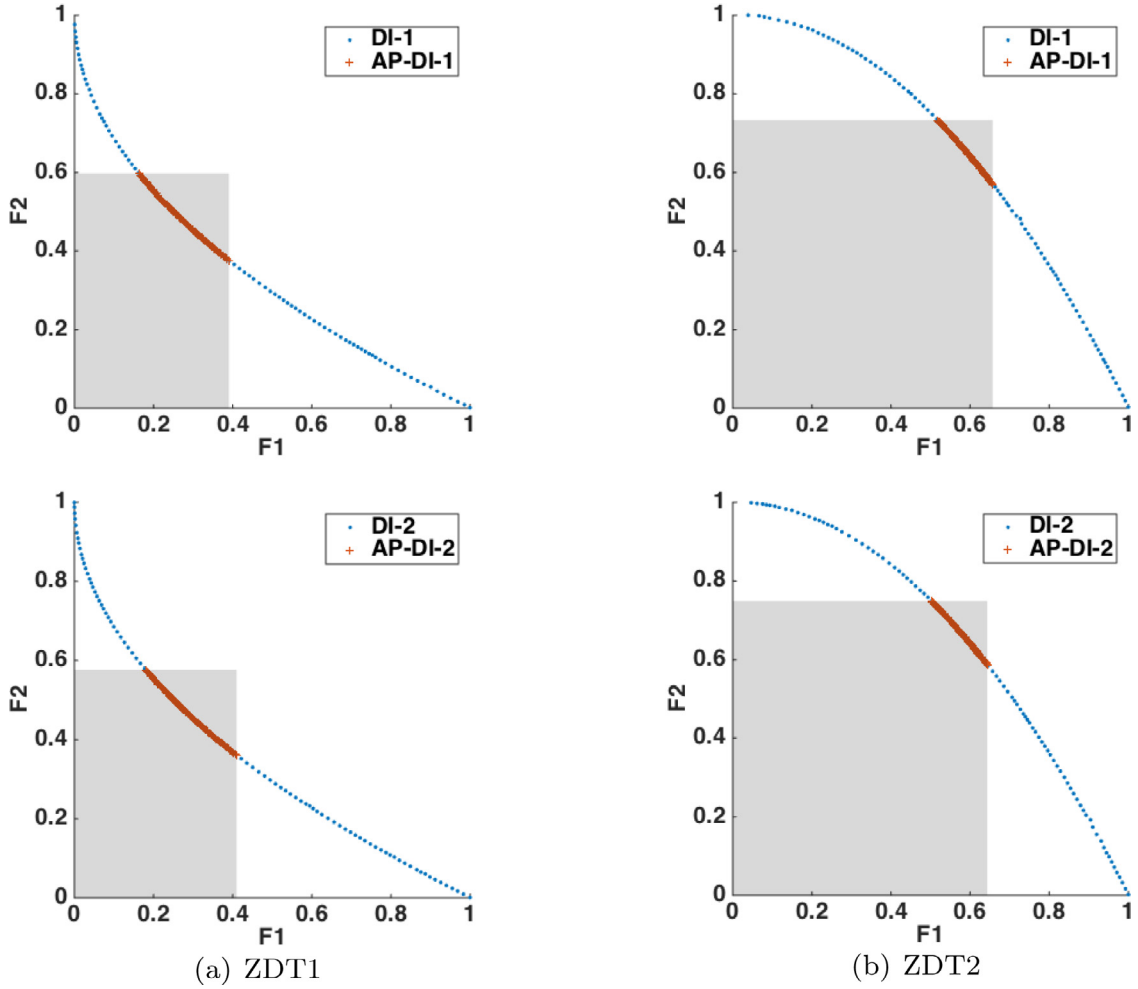


Fig. 5. Pareto front approximation on ZDT1 and ZDT2.

Table 1
Space and dominance relation of knee point from DI-MOEA and AP solutions on ZDT problems.

Problem		ZDT1		ZDT2	
Algorithm		DI-1/ AP-DI-1	DI-2/ AP-DI-2	DI-1/ AP-DI-1	DI-2/ AP-DI-2
In	Incomparable	20	23	1	1
preference	Dominated	10	7	9	9
region	Dominating	0	0	0	0
Outside	Incomparable	0	0	20	20
p-region	Dominated	0	0	0	0

Table 2
Space and dominance relation of knee point from NSGA-III and AP solutions on ZDT problems.

Problem		ZDT1		ZDT2	
Algorithm		NSGA-III/ AP-DI-1	NSGA-III/ AP-DI-2	NSGA-III/ AP-DI-1	NSGA-III/ AP-DI-2
In	Incomparable	14	19	3	1
preference	Dominated	0	0	2	3
region	Dominating	16	11	4	6
Outside	Incomparable	0	0	21	20
p-region	Dominated	0	0	0	0

and only solutions in the left side of the gray area can dominate AP solutions.

We also perform the same comparison between AP-DI-MOEA and NSGA-III, the results are shown in Table 2. For ZDT1 problem, all knee points from NSGA-III are in the preference regions from AP-DI-MOEA. Some of these knee points dominate AP solutions. For ZDT2 problem, most knee points from NSGA-III are not in the preference regions and these knee points are incomparable with AP solutions. For the knee points in the preference regions, all three dominating relations with AP solutions appear. For both problems, when the knee point from NSGA-III is dominating AP solutions, it only dominates one AP solution.

Instead of spreading the population across the entire Pareto front, we only focus on the preference region. To ensure that our algorithm can guide the search towards the preference region and the achieved solution set is distributed across the preference region, we compare the performance of AP-DI-MOEA, DI-MOEA and NSGA-III in the preference region. For each Pareto front approximation from DI-MOEA and NSGA-III, the solutions in the corresponding preference region from AP-DI-MOEA are picked, and we compare these solutions with AP solutions through the hypervolume indicator. The point formed by the largest objective values over all solutions in the preference region is adopted as the reference point when calculating the hypervolume indicator. It has been found that all hypervolume values of new solution sets from DI-MOEA and NSGA-III in the preference region are worse than the hypervolume values of the solution sets from AP-DI-MOEA, which proves that the mechanism indeed works in practice. Fig. 6 shows box plots of the distribution of hypervolume indicators over 30 runs.

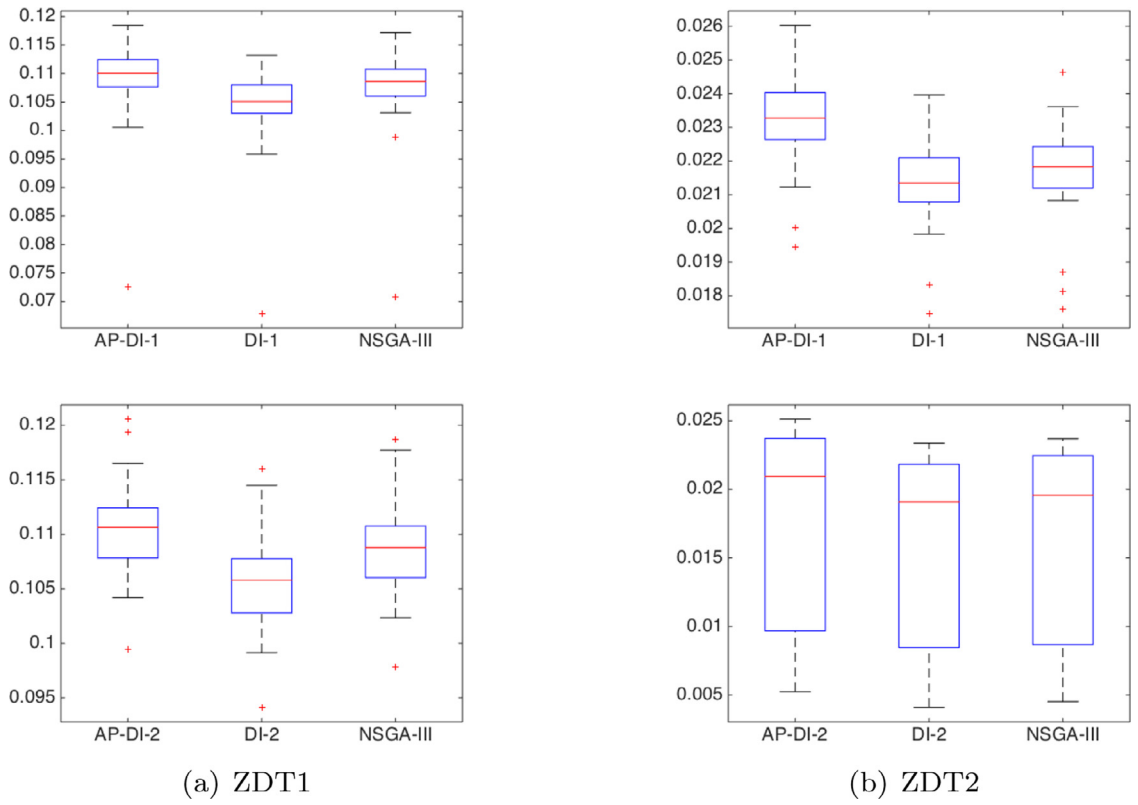


Fig. 6. Boxplots comparing the hypervolume values on ZDT1 and ZDT2.

Table 3
Space and dominance relation of knee point from DI-MOEA and AP solutions on DTLZ problems.

Problem		DTLZ1		DTLZ2	
Algorithm		DI-1/ AP-DI-1	DI-2/ AP-DI-2	DI-1/ AP-DI-1	DI-2/ AP-DI-2
In	Incomparable	29	27	10	13
preference	Dominated	0	0	1	0
region	Dominating	0	0	0	0
Outside	Incomparable	1	3	19	17
p-region	Dominated	0	0	0	0

Table 4
Space and dominance relation of knee point from NSGA-III and AP solutions on DTLZ problems.

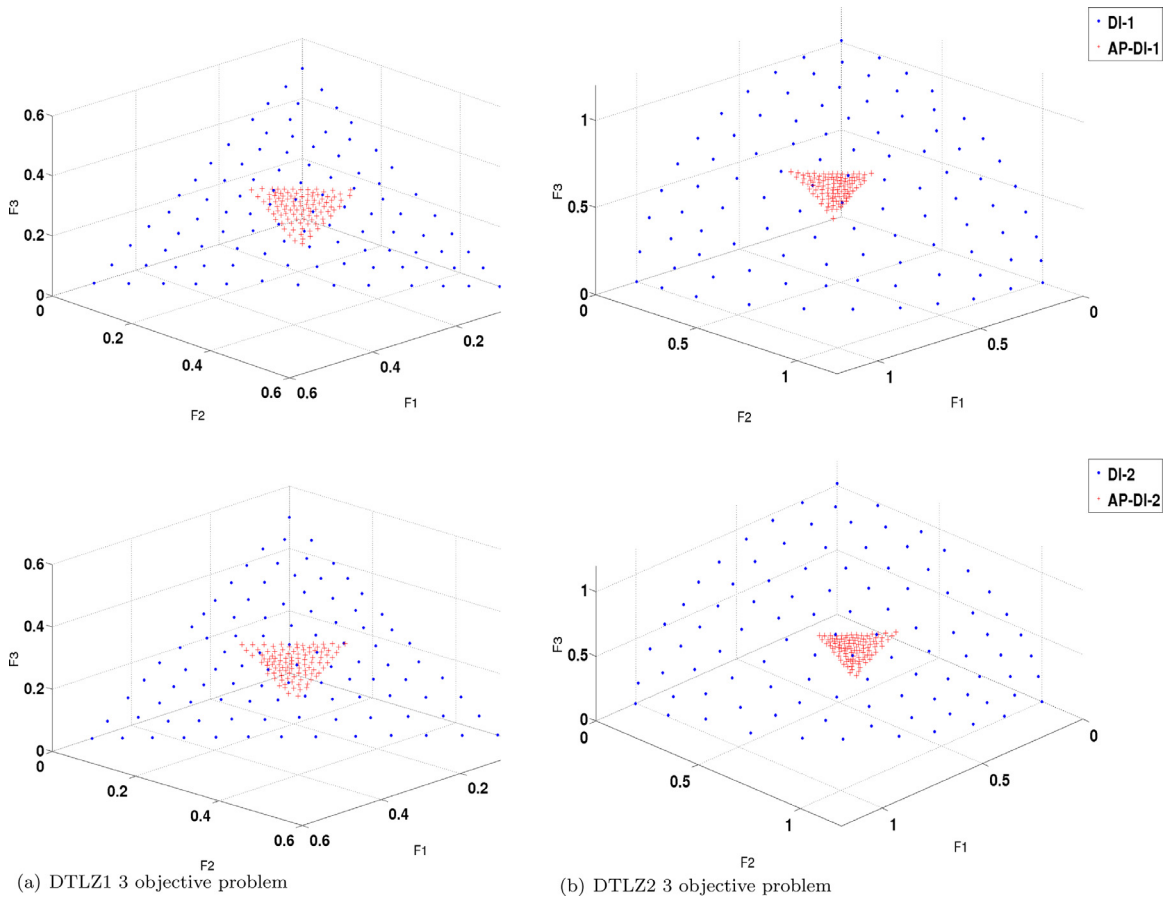
Problem		DTLZ1		DTLZ2	
Algorithm		NSGA-III/ AP-DI-1	NSGA-III/ AP-DI-2	NSGA-III/ AP-DI-1	NSGA-III/ AP-DI-2
In	Incomparable	30	29	14	17
preference	Dominated	0	0	1	1
region	Dominating	0	0	0	2
Outside	Incomparable	0	1	15	10
p-region	Dominated	0	0	0	0

6.3. *Experiments on three objective problems*

DTLZ1 and DTLZ2 are chosen as three objective benchmark problems to investigate our algorithms. They are performed with a total budget of 120000 fitness evaluations, when the evaluation reaches $60,000$ times, the first preference region is formed, then after every 5000 evaluations, the preference region is updated. Fig. 7 shows the Pareto front approximations from a typical run on DTLZ1 (left column) and DTLZ2 (right column). The upper graphs are obtained from DI-1 and AP-DI-1, while the lower graphs are from DI-2 and AP-DI-2. In each graph, the Pareto front approximations from DI-MOEA and corresponding AP-DI-MOEA are given. Since the target region is actually an axis aligned box, the obtained knee region (i.e., the intersection of the axis aligned box with the Pareto front) has an inverted triangle shape for these two benchmark problems.

Table 3 shows the space and dominance relation of the knee point from DI-MOEA and the solution set from AP-DI-MOEA over 30 runs. For DTLZ1 problem, most knee points from DI-MOEA are in their respective preference regions and all knee points are mutually non-dominated with AP solutions. For DTLZ2 problem, we observed that more knee points are not in the corresponding preference regions. This is because too few solutions from DI-MOEA are in the preference region. For DTLZ1 problem, six solutions from DI-MOEA are in the corresponding preference region on average for each run, while, for DTLZ2 problem, only less than two solutions are in the corresponding preference region on average. Therefore, we can see that on the one side, it is normal that many knee points from the entire Pareto fronts are not in their corresponding preference regions; on the other side, our aim of finding more fine-grained resolution in the preference region has been well achieved because only few solutions can be obtained in the preference region if we spread the population across the entire Pareto front. At the same time, one knee point from DI-1 on DTLZ2 is dominated by solutions from the corresponding AP-DI-1, which proves that AP-DI-MOEA can converge better than DI-MOEA because AP-DI-MOEA focuses on the preference region.

AP-DI-1 and AP-DI-2 have also been compared with NSGA-III in the same way. Table 4 shows the comparison result. For DTLZ1, the average number of solutions from NSGA-III in the corresponding preference regions from AP-DI-MOEA is six. Still, almost all knee solutions from NSGA-III are in the preference region. For DTLZ2, the average number of solutions from NSGA-III in the corresponding preference region from



(a) DTLZ1 3 objective problem (b) DTLZ2 3 objective problem

Fig. 7. Pareto front approximation on DTLZ1 and DTLZ2.

AP-DI-MOEA is less than one, while, in more than half of 30 runs, the knee points from NSGA-III are still in the preference region. To some extent, it can be concluded that the preference regions from AP-DI-MOEA are accurate. It can also be observed that AP-DI-1 behaves better than AP-DI-2 on DTLZ2, because two knee points from NSGA-III dominate the solutions from AP-DI-2.

Similarly, we pick from DI-MOEA and NSGA-III solutions which are in the corresponding preference region of AP-DI-MOEA, and the hypervolume indicator value is compared between these solutions and AP solutions. It has been found that all hypervolume values of solutions from AP-DI-MOEA are better than those of solutions from DI-MOEA and NSGA-III. The left column of Fig. 8 shows box plots of the distribution of hypervolume values over 30 runs on DTLZ1, and the right column shows the hypervolume comparison on DTLZ2.

In our experiments, we decide half of the total budget is used to find an initial Pareto front because it turned out to be a good compromise: half budget for the initial Pareto front and another half budget for the solutions focusing on the preference region. We also run experiments using 25% and 75% of the total budget for the initial Pareto front. Fig. 9 presents the entire Pareto front from DI-MOEA and the Pareto front from AP-DI-MOEA with different budgets for the initial Pareto front. The left two images are on DTLZ1 and the right two images are on DTLZ2. The upper two images are from DI-1 and AP-DI-1; the lower two images are from DI-2 and AP-DI-2. In the legend labels, 50%, 25% and 75% indicate the budgets which are utilized to find the initial entire Pareto front. It can be observed that the preference region from AP-DI-MOEA with 50% of budget is located in a better position than with 25% and 75% budgets, and the position of the preference region from AP-DI-MOEA with 50% of budget is more stable. Therefore, in our algorithm, 50% of the budget is used before the generation of the preference region.

6.4. Experiments on vehicle fleet maintenance scheduling optimization

The budget of $1,200,000$ evaluations has been used on the real-world application problems, and $600,000$ of them are for the initial Pareto front. After that, the preference region is updated after every $50,000$ evaluations. The VFMSO problem has been tested with different sizes. Fig. 10 shows Pareto front approximations of a problem with 20 cars and 3 workshops (V1), and each car contains 13 components: one engine, four springs, four brakes and four tires [44]. It can be observed that AP-DI-1 and AP-DI-2 can zoom in the entire Pareto front and find solutions in the preference region, at the same time, both AP-DI-1 and AP-DI-2 converge better than their corresponding DI-1 and DI-2. A similar conclusion can be drawn from Pareto fronts approximations of the problem with 30 cars and 5 workshops (V2) in Fig. 11.

In Fig. 12, We put the Pareto front approximations from DI-MOEA, AP-DI-MOEA and NSGA-III on V1 (left) and V2 (right) together. The behaviours of DI-1, DI-2 and NSGA-III are similar on V1, so are the behaviours of AP-DI-1 and AP-DI-2 on this problem. While, DI-2 and AP-DI-2 converge better than DI-1 and AP-DI-1 on V2 problem. The behaviour of NSGA-III is between that of DI-1 and DI-2.

Table 5 gives the space and dominance relation of knee points from DI-MOEA and solutions from AP-DI-MOEA on these two VFMSO problems. For both problems, only few knee points from DI-MOEA are in the preference regions of AP-DI-MOEA, and the main reason is that the Pareto front of AP-DI-MOEA converges better than that of DI-MOEA, in some cases, the Pareto front of DI-MOEA cannot even reach the corresponding preference region. More importantly, it can be observed that most knee points from DI-MOEA, no matter whether in the preference region or outside of the preference region, are dominated by the solu-

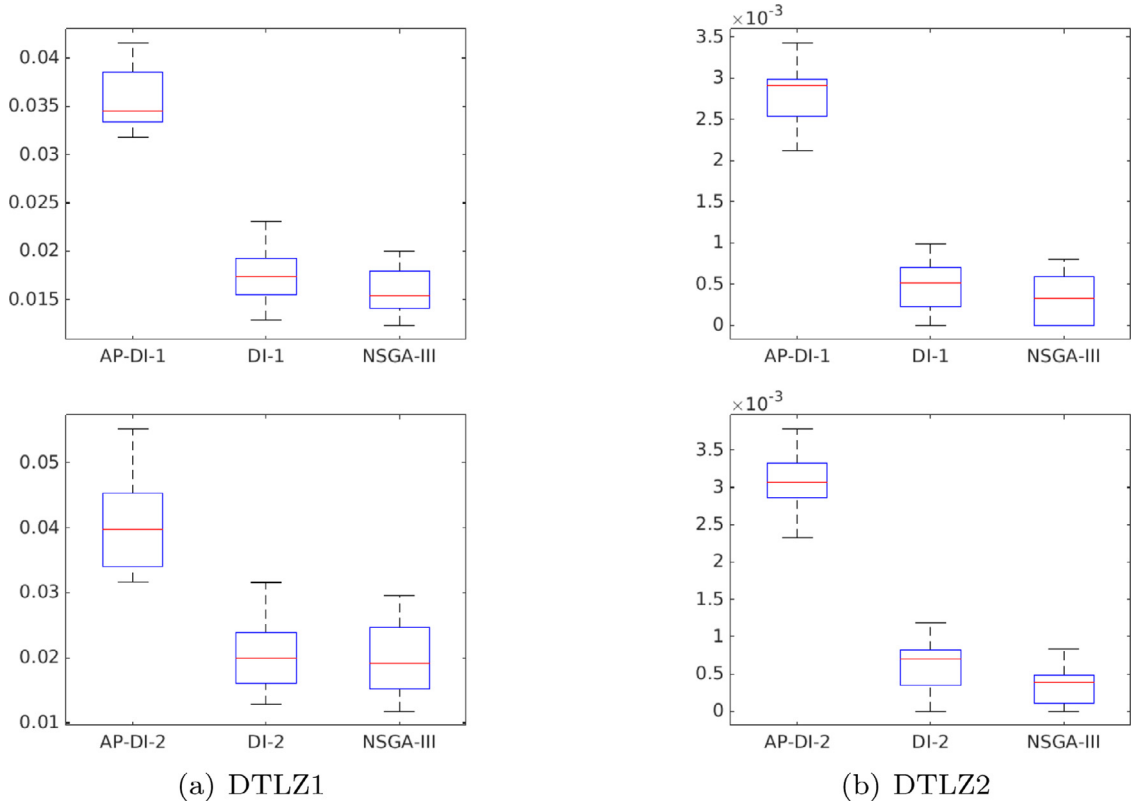


Fig. 8. Boxplots comparing the hypervolume values on DTLZ1 and DTLZ2.

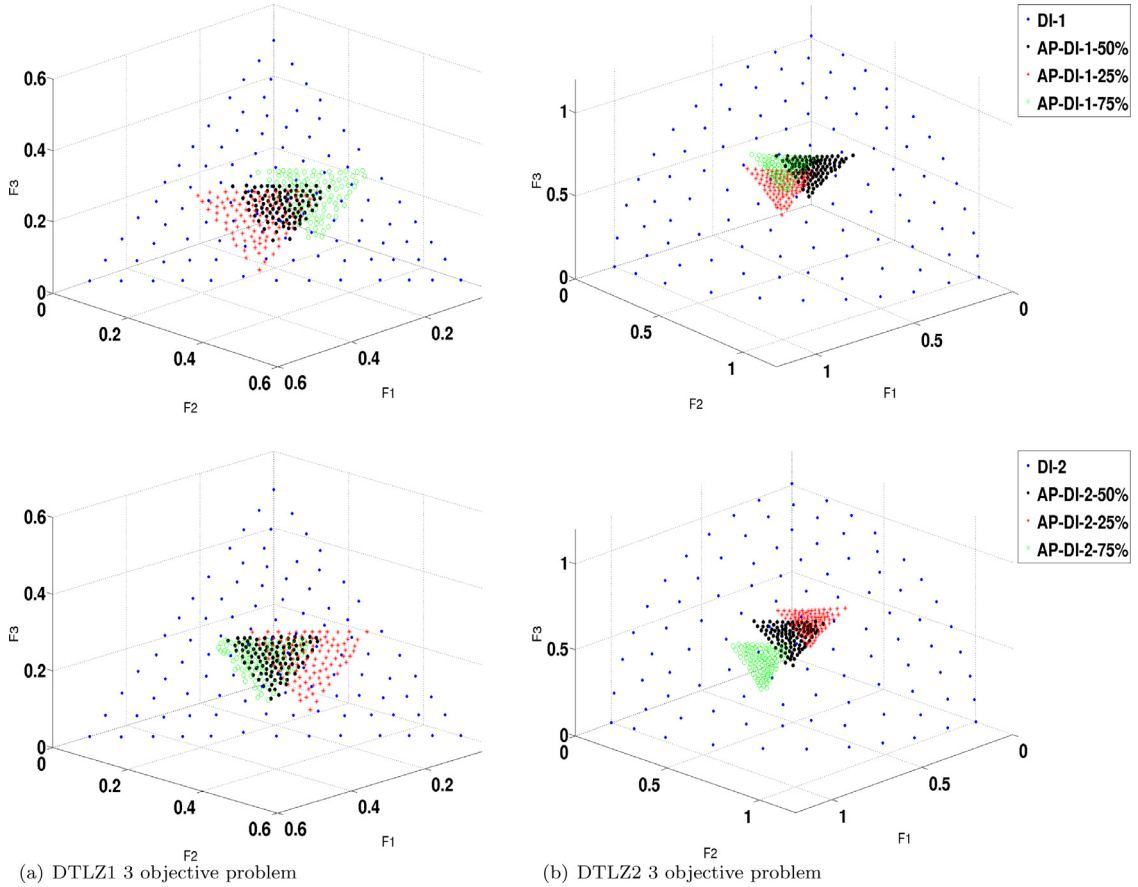
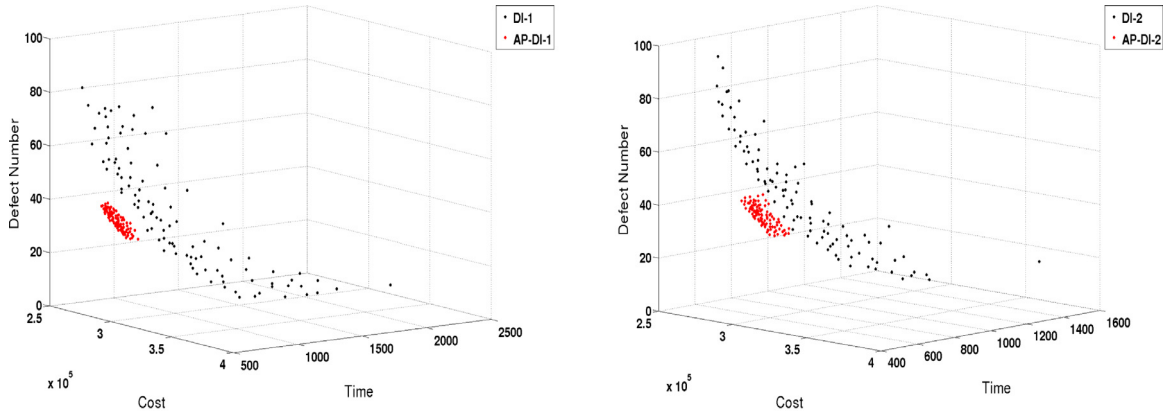
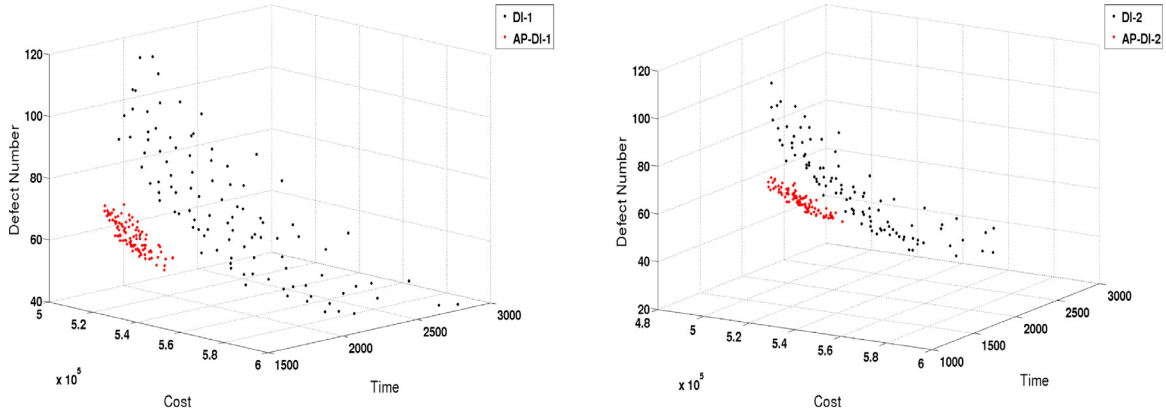


Fig. 9. Pareto front approximation by different budgets generating initial Pareto front.



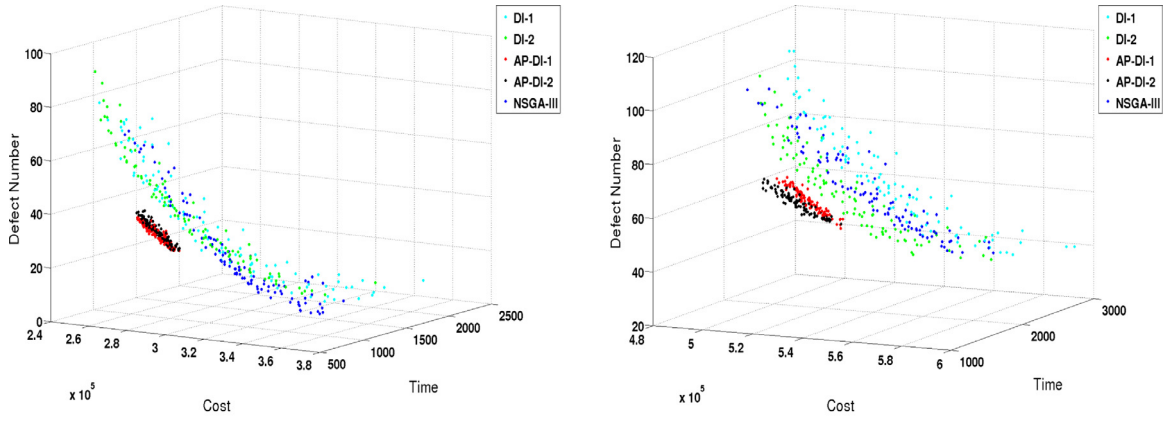
(a) DI-1 & AP-DI-1 (b) DI-2 & AP-DI-2

Fig. 10. Pareto front approximation on VFMSO problem with 20 cars and 3 workshops.



(a) DI-1 & AP-DI-1 (b) DI-2 & AP-DI-2

Fig. 11. Pareto front approximation on VFMSO problem with 30 cars and 5 workshops.



(a) V1 (b) V2

Fig. 12. Pareto front approximation on VFMSO problem by DI-MOEA, AP-DI-MOEA and NSGA-III.

tions from AP-DI-MOEA. This phenomenon is even more obvious for the application problem with bigger size and run with the same budget as the smaller one: for V2, 90% of knee points from DI-MOEA are dominated by the solutions from AP-DI-MOEA.

Table 6 gives the space and dominance relation of knee points from NSGA-III and AP solutions. For both problems, again, most knee points from NSGA-III are not in the preference regions of AP-DI-MOEA. Some knee points from NSGA-III are dominated by AP solutions and most of them are incomparable with AP solutions.

7. Conclusions

In this paper, a preference based multi-objective evolutionary algorithm, AP-DI-MOEA, is proposed. In the absence of explicitly provided preferences, the knee region is usually treated as the region of interest or preference region. Given this, AP-DI-MOEA can generate the knee region automatically and can find solutions with a more fine-grained resolution in the knee region. This has been demonstrated on the bi-objective problems ZDT1 and ZDT2, and the three objective problems

Table 5
Space and dominance relation of knee point from DI-MOEA and AP solutions on V1 and V2.

Problem		V1		V2	
Algorithm		DI-1/ AP-DI-1	DI-2/ AP-DI-2	DI-1/ AP-DI-1	DI-2/ AP-DI-2
In preference region	Incomparable	0	0	0	0
	Dominated	9	7	9	6
	Dominating	0	0	0	0
Outside p-region	Incomparable	4	9	3	3
	Dominated	17	14	18	21

Table 6
Space and dominance relation of knee point from NSGA-III and AP solutions on V1 and V2.

Problem		V1		V2	
Algorithm		NSGA-III/ AP-DI-1	NSGA-III/ AP-DI-2	NSGA-III/ AP-DI-1	NSGA-III/ AP-DI-2
In preference region	Incomparable	0	0	0	1
	Dominated	0	1	3	2
	Dominating	0	0	1	1
Outside p-region	Incomparable	23	24	21	18
	Dominated	7	5	5	8

DTLZ1 and DTLZ2. In the benchmark, the new approach was also proven to perform better than NSGA-III which was included in the benchmark as a state-of-the-art reference algorithm.

The research for the preference based algorithm was originally motivated by a real-world optimization problem, namely, Vehicle Fleet Maintenance Scheduling Optimization (VFMSO), which is described in this paper in a new formulation as a three objective discrete optimization problem. A customized set of operators (initialization, recombination, and mutation) is proposed for a multi-objective evolutionary algorithm with a selection strategy based on DI-MOEA and, respectively, AP-DI-MOEA. The experimental results of AP-DI-MOEA on two real-world application problem instances of different scales show that the newly proposed algorithm can generate preference regions automatically and it (in both cases) finds clearly better and more concentrated solution sets in the preference region than DI-MOEA. For completeness, it was also tested against NSGA-III and a better approximation in the preference region was observed by AP-DI-MOEA.

Since our real-world VFMSO problem is our core issue to be solved, and its Pareto front is convex, we did not consider problems with an irregular shape. It would be an interesting question how to adapt the algorithm to problems with more irregular shapes. Besides, the proposed approach requires a definition of knee points. Future work will provide a more detailed comparison of different variants of methods to generate knee points, as they are briefly introduced in Section 3. In the application of maintenance scheduling, it will also be important to integrate robustness and uncertainty in the problem definition. It is desirable to generate schedules that are robust within a reasonable range of disruptions and uncertainties such as machine breakdowns and processing time variability. Lastly, future work should also consider many-objective optimization problems which should be benefited by the optimization process which only focuses on part of the objective space.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yali Wang: Conceptualization, Formal analysis, Methodology, Software, Writing - original draft. **Steffen Limmer:** Data curation. **Markus Olhofer:** Project administration, Supervision. **Michael Emmerich:** Formal analysis, Supervision, Writing - original draft. **Thomas Bäck:** Project administration.

Acknowledgements

This work is part of the research programme Smart Industry SI2016 with project name CIMPLO and project number 15465, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

References

[1] Y. Wang, S. Limmer, M. Olhofer, M.T. Emmerich, T. Bäck, Vehicle fleet maintenance scheduling optimization by multi-objective evolutionary algorithms, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 442–449.
[2] H.M. Elattar, H.K. Elminir, A. Riad, Prognostics: a literature review, Complex Intell. Syst. 2 (2) (2016) 125–154.
[3] Y. Wang, M. Emmerich, A. Deutz, T. Bäck, Diversity-indicator based multi-objective evolutionary algorithm: DI-MOEA, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2019, pp. 346–358.
[4] A. Zakariazadeh, S. Jadid, P. Siano, Multi-objective scheduling of electric vehicles in smart distribution system, Energy Convers. Manag. 79 (2014) 43–53.
[5] F. Ramezani, J. Lu, J. Taheri, F.K. Hussain, Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments, World Wide Web 18 (6) (2015) 1737–1757.
[6] Y. Hou, N. Wu, M. Zhou, Z. Li, Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm, Ieee Trans. Syst. Man Cybern. 47 (3) (2015) 517–530.
[7] J.-Y. Ding, S. Song, C. Wu, Carbon-efficient scheduling of flow shops by multi-objective optimization, Eur. J. Oper. Res. 248 (3) (2016) 758–771.
[8] S.V. Jerić, J.R. Figueira, Multi-objective scheduling and a resource allocation problem in hospitals, J. Sched. 15 (5) (2012) 513–535.
[9] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, H.A. Chan, Multi-objective scheduling of micro-services for optimal service function chains, in: 2017 IEEE International Conference on Communications (ICC), IEEE, 2017, pp. 1–6.
[10] T.-C. Chiang, H.-J. Lin, A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling, Int. J. Prod. Econ. 141 (1) (2013) 87–98.
[11] Y. Yuan, H. Xu, Multiobjective flexible job shop scheduling using memetic algorithms, IEEE Trans. Autom. Sci. Eng. 12 (1) (2013) 336–353.
[12] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, Q. Pan, A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems, IEEE/CAA J. Autom. Sin. 6 (4) (2019) 904–916.
[13] C. Özgüven, L. Özbakır, Y. Yavuz, Mathematical models for job-shop scheduling problems with routing and process plan flexibility, Appl. Math. Model. 34 (6) (2010) 1539–1548.
[14] Y. Demir, S.K. İşleyen, An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations, Int. J. Prod. Res. 52 (13) (2014) 3905–3921.
[15] L. Yu, C. Zhu, J. Shi, W. Zhang, An extended flexible job shop scheduling model for flight deck scheduling with priority, parallel operations, and sequence flexibility, Sci. Program. (2017).
[16] P. Fattahi, A. Fallahi, Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability, CIRP J. Manuf. Sci. Technol. 2 (2) (2010) 114–123.
[17] N. Al-Hinai, T.Y. ElMekkawy, Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm, Int. J. Prod. Econ. 132 (2) (2011) 279–291.
[18] X.-N. Shen, X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Inf. Sci. 298 (2015) 198–224.
[19] E. Ahmadi, M. Zandieh, M. Farrokh, S.M. Emami, A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms, Comput. Oper. Res. 73 (2016) 56–66.
[20] I. Das, On characterizing the "knee" of the Pareto curve based on normal-boundary intersection, Struct. Optim. 18 (2–3) (1999) 107–115.
[21] C. Mattson, A. Mullur, A. Messac, Minimal representation of multiobjective design space using a smart Pareto filter, in: 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2002, p. 5458.
[22] K. Deb, Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions, in: Advances in Evolutionary Computing, Springer, 2003, pp. 263–292.
[23] J. Branke, K. Deb, H. Dierolf, M. Osswald, Finding knees in multi-objective optimization, in: International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 722–731.
[24] E. Zitzler, M. Laumanns, S. Bleuler, A tutorial on evolutionary multiobjective optimization, in: Metaheuristics for Multiobjective Optimisation, Springer, 2004, pp. 3–37.

[25] G. Yu, Y. Jin, M. Olhofer, A method for a posteriori identification of knee points based on solution density, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2018, pp. 1–8.
[26] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[27] L. Rachmawati, D. Srinivasan, A multi-objective evolutionary algorithm with weighted-sum niching for convergence on knee regions, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 749–750.
[28] L. Rachmawati, D. Srinivasan, A multi-objective genetic algorithm with controllable convergence on knee regions, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 1916–1923.
[29] O. Schütze, M. Laumanns, C.A.C. Coello, Approximating the knee of an mop with stochastic search algorithms, in: International Conference on Parallel Problem Solving from Nature, Springer, 2008, pp. 795–804.
[30] K. Deb, S. Gupta, Understanding knee points in bicriteria problems and their implications as preferred solution principles, Eng. Optim. 43 (11) (2011) 1175–1204.
[31] K. Deb, J. Sundar, Reference point based multi-objective optimization using evolutionary algorithms, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 635–642.
[32] Y. Vesikar, K. Deb, J. Blank, Reference point based NSGA-III for preferred solutions, in: 2018 IEEE Symposium Series on Computational intelligence (SSCI), IEEE, 2018, pp. 1587–1594.
[33] S. Bechikh, L. Ben Said, K. Ghédira, Searching for knee regions in multi-objective optimization using mobile reference points, in: Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 1118–1125.
[34] D. Gaudrie, R. Le Riche, V. Picheny, B. Enaux, V. Herbert, Targeting solutions in Bayesian multi-objective optimization: sequential and batch versions, Ann. Math. Artif. Intell. (2019) 1–26.
[35] L. Rachmawati, D. Srinivasan, Multiobjective evolutionary algorithm with controllable focus on the knees of the Pareto front, IEEE Trans. Evol. Comput. 13 (4) (2009) 810–824.
[36] L. Thiele, K. Miettinen, P.J. Korhonen, J. Molina, A preference-based evolutionary algorithm for multi-objective optimization, Evol. Comput. 17 (3) (2009) 411–436.
[37] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 832–842.
[38] M.A. Braun, P.K. Shukla, H. Schmeck, Preference ranking schemes in multi-objective evolutionary algorithms, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2011, pp. 226–240.
[39] C. Ramirez-Atencia, S. Mostaghim, D. Camacho, A knee point based evolutionary multi-objective optimization for mission planning problems, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2017, pp. 1216–1223.
[40] Y. Wang, L. Li, K. Yang, M.T. Emmerich, A new approach to target region based multiobjective evolutionary algorithms, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1757–1764.
[41] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: multiobjective selection based on dominated hypervolume, Eur. J. Oper. Res. 181 (3) (2007) 1653–1669.
[42] M. Pal, S. Saha, S. Bandyopadhyay, Decor: differential evolution using clustering based objective reduction for many-objective optimization, Inf. Sci. 423 (2018) 200–218.
[43] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints, IEEE Trans. Evol. Comput. 18 (4) (2013) 577–601.
[44] D. Van Nguyen, S. Limmer, K. Yang, M. Olhofer, T. Bäck, Modeling and prediction of remaining useful lifetime for maintenance scheduling optimization of a car fleet., Int. J. Perform. Engineering 15 (9) (2019).