Chapter 6

Preference-based and Dynamic Vehicle Fleet Maintenance Scheduling Optimization

The first version of the multi-objective vehicle fleet maintenance scheduling optimization problem has been formulated and solved by the proposed algorithms in the previous chapter. To make the problem more practical, rigorous and clear, after discussing with the DM from Honda Research Institute Europe GmbH, the problem is upgraded from the following aspects:

- There exists a lot of uncertainty when the predicted RUL of each component is used 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 using the predicted RUL, predicted RUL probability distribution should be used as the foundation to assign the maintenance time in scheduling optimization.
- The expected number of failures is adopted as an objective to reduce the chances that the vehicles are broken on the road.
- The teams in workshops don't need to be specified, each workshop can be treated as one team.

- The demand satisfaction is removed from objectives. It is not necessary to consider it for a more general problem.

To implement these changes, the MOVFMSO problem is reformulated in this section. Naturally, the corresponding MOEAs need to be modified to solve the newly formulated problem. These are all described in Section 6.1. Besides, AP-DI-MOEA (Automatic Preference based DI-MOEA) is also adopted in Section 6.1 to find solutions with a more fine-grained resolution in the automatically generated preference region.

To model the complete process of the vehicle fleet maintenance scheduling optimization, a VFMSO simulator is developed in Section 6.2. The VFMSO simulator starts from simulating driving tasks and available workshops for a vehicle fleet. The RULs of components are predicted when the vehicles execute the distributed driving tasks. Afterwards, the proposed MOEAs are applied to optimize the maintenance schedule, and the workshops can maintain the vehicles based on the optimal schedule. The process is running in a rolling-horizon fashion and a new maintenance schedule is generated periodically based on the newly predicted RULs. To do this, a fourth objective is added into the optimization, which is to minimize the changes between the new schedule and the previous schedule. Thus, the optimization algorithms are extended to dynamic MOEAs.

6.1 Preference-base MOEAs for MOVFMSO

This section starts with the new formulation of the multi-objective vehicle fleet maintenance scheduling optimization problem in Section 6.1.1. The tailored algorithm to solve the new optimization problem is described in Section 6.1.2. The performance of MOEAs and preference-based MOEAs on the problem are reported in Section 6.1.3. Lastly, Section 6.1.4 concludes the work and outlines directions for future work.

6.1.1 Problem Formulation

For a vehicle fleet running the driving tasks, the components of vehicles are getting damaged 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. Besides the time and cost for repairing the car component, a fixed set-up cost and set-up time are 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 updated VFMSO problem addressed in this section is defined as follows:

1. There are n cars $C = \{C_1, C_2, \cdots, C_n\}$ and m workshops $W = \{W_1, W_2, \cdots, W_m\}$.
2. Each car C_i comprises l_i components to be maintained for $i = 1, \cdots, n$.
3. For each component O_{ij} $(j = 1, \cdots, l_i)$, 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 previous repair time of component O_{ij} is recorded and denoted by L_{ij}.

The constraint in this problem 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.

Three objectives are taken into consideration, 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 this problem, the fact that 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:

Minimize the total workload: $f_1 = \sum_{k=1}^{m} T_k$ (6.1)

Minimize the total cost: $f_2 = \sum_{i=1}^{n} M_i$ (6.2)

Minimize the expected number of failures:

$$f_3 = \sum_{i=1}^{n} \sum_{j=1}^{l_i} \mathbb{E}(F_{ij}). \tag{6.3}$$

6.1.2 Customized Algorithm

First the execution window is defined 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.

After the determination of the execution window, the maintenance of several components can be combined 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. This part is the same as described in Section 5.2.3.

Within the execution window of a component, an arbitrary time can be chosen as the starting time for maintaining the component. However, the maintenance time of each component should be as close as possible to its real due date to save its useful time and avoid a car breakdown on the road. Therefore, Monte Carlo simulation is used to simulate the "real" due dates for each component. To be specific, 1000 samples of the due date are generated in the execution window of each component according to its predicted RUL probability distribution. Figure 6.1 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:

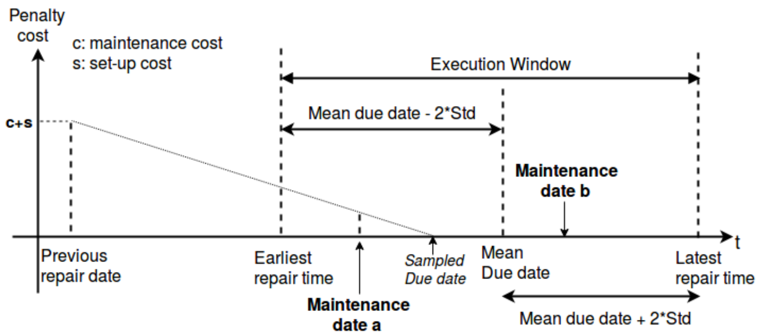


Figure 6.1: Execution window of a component.

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 Figure 6.1, 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 Figure 6.1, but the scheduled maintenance date D_{ij} is "Maintenance date b". In this case, D_{ij} is later than d^v_{ij}, and the vehicle will break down on the road. In the algorithm, the number of failures will be increased by one.

Case 3 $D_{ij} = d^v_{ij}$

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.

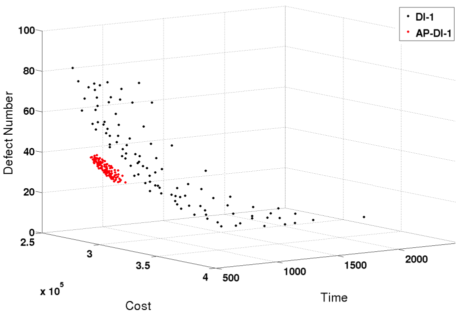
The averages of the penalty costs and the number of failures from 1000 due date samples will be used as the penalty cost and expected number of failures for the scheduled maintenance date of the component. For each operation (the single-component operation or group operation), its cost consists of three 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 the multi-objective optimization.

In Section 5.2.3, the implementation of tailored evolutionary algorithm for the first formulation of the MOVFMSO problem has been introduced, including how to represent an individual or solution in the population, how to take these chromosomes into a process of evolution, how to create variations of solutions in each iteration, etc. The algorithm can still be used on the updated problem. Next, AP-DI-MOEA (described in Section 4.2.2) is conducted on the updated MOVFMSO problems to demonstrate the performance.

6.1.3 Experimental Results

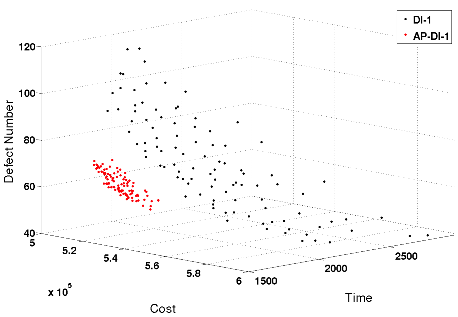
The performance of MOEAs and the preference-based MOEAs are compared on the VFMSO problems. The two variants of AP-DI-MOEA: AP-DI-1 and AP-DI-2, have been conducted on two instances with different sizes. On every problem, each algorithm runs 30 times with different seeds, while the same 30 different seeds are used for all algorithms. All the experiments are performed with a population size of 100. The budget of 1200000 evaluations has been used and 600000 of them are for the initial Pareto front; after that, the preference region is updated after every 50000 evaluations.

Figure 6.2 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 [109]. 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 Figure 6.3.



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

Figure 6.2: Pareto front approximation on VFMSO problems with 20 cars and 3 workshops.

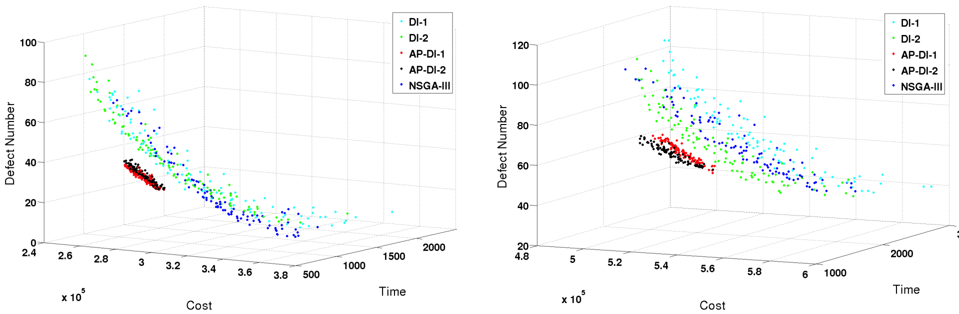


(a) DI-1 & AP-DI-1. **(b)** DI-2 & AP-DI-2.

Figure 6.3: Pareto front approximation on VFMSO problems with 30 cars and 5 workshops.

In Figure 6.4, the Pareto front approximations from DI-MOEA, AP-DI-MOEA and NSGA-III on V1 (left) and V2 (right) are put 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 problems. The behaviour of NSGA-III is between that of DI-1 and DI-2.

Table 6.1 gives the space and dominance relation of knee points from DI-MOEA



(a) V1. **(b)** V2.

Figure 6.4: Pareto front approximation on VFMSO problems by DI-MOEA, AP-DI-MOEA and NSGA-III.

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 solutions 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 6.1: 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 6.2 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.

Table 6.2: 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

6.1.4 Conclusion

The multi-objective vehicle fleet maintenance scheduling optimization problems were updated after further discussion with the decision makers. In the new optimization problem, the maintenance time of each component was based on the predicted distribution of its remaining useful time. A new objective, i.e., the expected number of failures, was adopted to reduce the risk of car breakdown on the road.

The proposed MOEAs and preference-based MOEAs have been conducted on the updated MOVFMSO problems. The experimental results of AP-DI-MOEA on two application problem instances of different scales showed that AP-DI-MOEA can generate preference regions automatically and it (in both cases) found 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.

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.

6.2 Dynamic MOEAs for MOVFMSO

Up to this point, the real-world application problem, i.e., the vehicle fleet maintenance scheduling optimization, has been formulated; the tailored multi-objective evolutionary algorithms have been developed; the basic MOEAs have been extended to the preference based MOEAs for the VFMSO problems. So far these proposed algorithms are used to solve the static problems. However, in the real-world scenario, after a maintenance schedule is released for execution, continuously updating the schedule is required due to the change of vehicle conditions and the ensuing changes in the

RUL predictions. The optimization of the maintenance schedule is an ongoing process running in a rolling-horizon fashion and it is therefore desirable to generate robust schedules.

According to the literature in robust scheduling methodologies, robustness is mainly grouped into quality robustness and solution robustness [57]. The quality robustness refers to the insensitivity of the scheduling performance such as makespan and total tardiness in the presence of uncertainty. The property that the start and the completion of each activity should be as close as possible to its previous schedule is known as the solution robustness and it is usually considered as a stability measurement of the schedule. When the proposed static algorithm is extended to a dynamic algorithm, a fourth objective is involved in the algorithm, which is the stability, i.e., the solution robustness.

To model the complete process of maintaining the vehicle fleet by way of scheduling optimization, a simulator is developed to observe the performance of dynamic MOEAs. The VFMSO simulator starts from simulating driving tasks and available workshops for a vehicle fleet, at the same time, in the simulator, the RUL of components can be predicted and used as the input information to optimize the maintenance schedule for the vehicle fleet. During the running of the simulator, the optimization process is running in a rolling-horizon fashion and the maintenance schedule is updated periodically. Accordingly, the optimization algorithm becomes a dynamic algorithm and a fourth objective is added into the dynamic MOEA, which is to minimize the changes between the new schedule and the previous schedule.

This section first introduces dynamic optimization for the VFMSO problems in Section 6.2.1. The RUL prediction is described in Section 6.2.2. Section 6.2.3 discusses the details of the simulator and Section 6.2.4 shows the experimental results. Finally, Section 6.2.5 briefly summarises the study, and proposes possible directions for future work.

6.2.1 Dynamic Optimization

By applying the proposed MOEA and preference-based MOEA, namely DI-MOEA and AP-DI-MOEA, after achieving a PF approximation, the knee point is picked as the final optimal schedule to be deployed in workshops. In the real-world application, the maintenance schedule needs to be updated periodically. To generate a new schedule for the next stage, the current schedule used for the vehicle fleet and workshops is also needed. Various disruptions may occur while running a maintenance schedule, for example, the car is broken before its scheduled maintenance time, or new repairing tasks in workshops lead to the delay of the scheduled activities. In the face of various disruptions, adjustments in the schedule have to be made and this is also the reason that the maintenance schedule is updated periodically. A new schedule with the new arrangement of the maintenance activities is generated from the new condition of the vehicle fleet and workshops. However, the changes on the current schedule lead to additional costs such as the cost of reallocation of tools and equipment, the cost of reordering of raw materials, and etc. To reduce these costs, when updating the maintenance schedule, one important point is to maximize the similarity between the new schedule and the previous one to increase stability. For this purpose, the stability criterion is employed as one more objective in the dynamic algorithm. Let $d_{ij} = 1$ if the maintenance time or workshop of component O_{ij} in the previous optimal maintenance schedule is different with the assigned maintenance time or workshop of component O_{ij} in the currently optimized schedule, otherwise $d_{ij} = 0$. The stability of the schedule can be maximized by minimizing the difference in schedule.

$$\text{Minimize the schedule difference:} \quad f_4 = \sum_{i=1}^{n} \sum_{j=1}^{l_i} d_{ij}. \tag{6.4}$$

The number of components which are assigned to different maintenance times or workshops from that in the current running schedule is minimized in the dynamic algorithm. Furthermore, since the stability of maintenance activities in the near future is more important than that of maintenance activities in the distant future, when calculating the stability objective, different weights are given to the components which are scheduled to be maintained within one week, within one month and beyond one month. The dynamic algorithm makes it possible that the maintenance schedule is optimized under different operational environments including dynamic and changing conditions. Most importantly, the dynamic algorithm updates the maintenance schedule based on the latest damage of components because the underlying predicted RUL of each component is based on the latest damage. In this way, the maintenance schedule becomes more accurate.

6.2.2 Remaining Useful Lifetime Prediction

Knowing the RUL is essential to establish an optimal maintenance schedule, and the RUL prediction provides the system residual life from its current condition and the past operation profile [106]. Commonly, approaches used in prognostics and predicting RUL are classified into three types: physics-based approaches, data-driven approaches, and hybrid approaches [35]. In this work, the physics-based models are used to estimate the degradation and failures of four essential components of a vehicle, namely engine, brake pads, helical springs, and tires. The fatigue and wear mechanisms are established for these components.

Degradation of Helical Springs

Helical spring is the most common type of spring used in passenger cars. One of the the main mechanisms that reduces the lifetime of a helical spring is fatigue and it is often analyzed using the S-N curve which describes the relation between cyclic stress amplitude and number of cycles to failure. Figure 6.5 shows a typical S-N curve. The vertical axis shows the stress amplitude, whereas the horizontal axis indicates the corresponding number of cycles to failure at a given stress amplitude. A stress S is calculated from force F by the equation: $S = K\frac{8\times F\times D_{coil}}{\pi\times d_{wire}^3}$, where D_{coil} and d_{wire} are the diameter of the mean coil and the wire, respectively. $C = \frac{D_{coil}}{d_{wire}}$ is the spring index. $K = 1 + \frac{0.5}{C}$ is the so-called Wahl factor. According to the Paris-Erdogan's and Palmgren-Miner laws [97], the damage percentage of a spring can be formulated as: $d_s = \sum_{i=1}^{p} \frac{n_i}{N_i} \times 100\%$, where d_s is the total percentage of life consumed, p is the total number of the considered stress sources, n_i and N_i are the number of cycles with a stress amplitude and the corresponding number of cycles to failure at this stress with $i = 1, 2, ..., p$ from p sources. $\frac{n_i}{N_i}$ is the fractional damage received from the ith source. When $d_s \geq 100\%$, the spring's lifetime ends and a spring failure occurs.

Degradation of Brake Pads

A wear-out failure arises as a result of cumulative damage related to loads applied over an extended time. In the process of braking, due to friction between the surfaces of the friction couple, the zones of contacts are damaged after each braking event, resulting in worn-out material. The volume of the worn-out material of the ith braking event can be represented as: $\triangle V_{bi} = C_{brake} \times F_i \times \triangle d_i$, where C_{brake} is a constant and presents the brake pad quality, F_i and $\triangle d_i$ are the friction force and the relative displacement between the brake pad and the brake rotor of ith braking event, respectively. If V_{b0} is the maximum volume which the brake pad can reduce before a failure might occur, damage percentage of the brake pad (d_b) can be estimated by $d_b = \sum_{i=1}^{n} \frac{\triangle V_{bi}}{V_{b0}} \times 100\%$. The brake force is converted from the brake torque by dividing torque by the length of the level arm. For the values of parameters in the physical models, such as D_{coil},

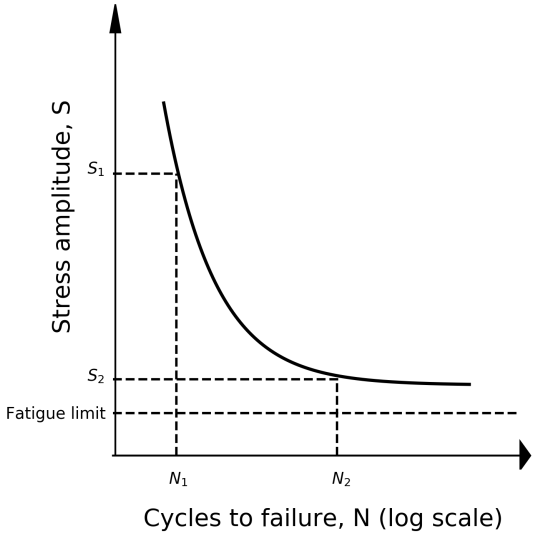


Figure 6.5: A typical S-N curve.

d_{wire}, C_{brake}, C_{tire}, C_{engine} (please refer to [109]).

Degradation of Tires

The wear mechanism is also applied to tires because tires' surfaces are in contact with the road surface and friction results in worn-out material of the tires. Two horizontal components of the force that cause the tire worn-out are F_x and F_y. The vertical force component F_z is only considered for pressure (overinflation, underinflation) damage of the tires. Similarly, a volume reduction of the tire due to worn-out material is formulated as: $\triangle V_{ti} = C_{tire} \times (|F_x| + |F_y|) \times \triangle d_i$, where C_{tire} is a constant and represents the tire quality. $\triangle d_i$ is the relative displacement between the tire surface and the road surface and it is simply the car travel distance. Again, the damage percentage of the tire (d_t) can be computed by: $d_t = \sum_{i=1}^{n} \frac{\triangle V_{ti}}{V_{t0}} \times 100\%$, where V_{t0} is the maximum volume which the tire can reduce before a failure might occur.

Degradation of Car Engine

A rough model is established to estimate the consumption lifetime of the car engine from the travel distance and the engine rotation speed. The equation is $d_{ei} = C_{engine} \times \triangle d_i \times R_i$, where C_{engine} is a constant and represents the engine quality. Here $\triangle d_i$

and R_i are the car travel interval, and the engine rotation speed corresponding to this travel interval, respectively. The consumed lifetime percentage of the engine is $d_e = \sum_{i=1}^{n} d_{ei} \times 100\%$. The engine needs to be maintained if the d_{ei} sum up to 1.

RUL Calculation

It is assumed that the physical models are accurate, therefore, the real damage on components up to now can be diagnosed. The RUL is predicted by extrapolating the future damage from the distribution of the damage so far. The RUL of a component can be calculated based on a damage percentage. If the RUL is estimated by a unit of week, the total damage percentage after the wth week is calculated by: $D = \sum_{i=1}^{w} D_i$, where D_i is the sum damage percentage of the ith week. Thus, the RUL after week w can be estimated by: $RUL = \frac{100\%-D}{D/w}$, here, 100% means that, at the beginning, the component is absolutely new. A Gaussian distribution is fitted to the distribution of the weekly damage percentage and the resulting standard deviation σ is used to calculate the lower and upper bound of the standard deviation confidence interval of RUL as following: $RUL_{-} = \frac{100\%-D}{D/w+\sigma}$ and $RUL_{+} = \frac{100\%-D}{D/w-\sigma}$.

6.2.3 VFMSO Simulator

A simulator has been developed to implement and evaluate the complete process of vehicle fleet maintenance scheduling optimization. In the VFMSO simulator, CarMaker[1] is adopted to simulate driving scenarios for a taxi fleet in New York City. The origin and destination coordinates from Green Taxi Company in January 2015 downloaded from NYC Open Data[2] are converted into taxi routes using Google API and are used as the driving tasks. In the CarMaker simulation, extra loads are added to all passenger seats of the car. For each passenger seat a load between 0 and 100 kg is randomly chosen with an equal probability. 4000 trips have been simulated with CarMaker. In the VFMSO simulator, each car is assigned to 40 random trips per day on average, and the maximum number of trips each car can execute per day is 50. These trips are randomly selected from the 4000 simulated trips. The sensor data of forces, brake torque and engine rotation speed yielded by CarMaker are used to estimate the damage percentage and the RUL of springs, tires, brake pads and engine

[1]CarMaker simulation is developed by IPG Automotive for testing driving scenarios of passenger cars and light-duty vehicles. It provides models for vehicles, roads, drivers and traffic for all simulation tasks in realistic driving scenarios. https://ipg-automotive.com/products-services/simulation-software/carmaker/#driver

[2]https://data.cityofnewyork.us/Transportation/2015-Green-Taxi-Trip-Data/gi8d-wdg5/data

by the physical models as described in Section 6.2.2.

Some parameters can be pre-defined to determine the vehicle fleet maintenance scheduling optimization problem before running the simulator: the size of the vehicle fleet, including the number of cars and the number of workshops; the costs and times of maintaining cars/components in workshops; the range of days of running the simulator; the frequency of generating a new maintenance schedule. After running the simulator for the defined period, the following items can be reported by the simulator.

- The number of defects: when a defect occurs, i.e., a component is broken before the scheduled maintenance date, the number of defects increases by one.
- The total cost: besides the set-up cost and maintenance cost, the waste of component lifetime has also been transferred to a cost, and has been included in the total cost by the simulator, this cost is called "too-early maintenance cost". Unlike the penalty cost in the optimization algorithm, the "too-early maintenance cost" in the simulator is the actual value because it is assumed that the physical models are 100% accurate and the due dates of the components calculated by them are used as the ground truth. When a component is maintained based on the maintenance schedule, the simulator can calculate its current damage percentage by the corresponding physical model and the remaining damage percentage is converted to a cost to reflect the waste of the useful lifetime. The "too-early maintenance cost" is calculated by the formula: remaining damage percentage $\times$ maintenance cost of the component. Obviously, no "too-early maintenance cost" arises for components which break before maintenance.
- The total maintenance time: the simulator records all the days that the vehicles cannot work, either the reason is a scheduled maintenance activity or a defect.
- The number of changed schedules: every time when the maintenance schedule is updated, the number of components which have a different maintenance date or workshop is recorded.
- The number of unsatisfied trips: when a car cannot execute its tasks, e.g., it is being maintained in a workshop, the assigned tasks for this car will be distributed to other available cars, but the maximum number of tasks a car can execute each day is 50. The tasks which cannot be satisfied are counted as unsatisfied trips.
- The number of scheduled maintenance activities: when a maintenance activity is executed based on the maintenance schedule, the number of scheduled maintenance activities increases by one.

These items are the final results after running the simulator for a pre-defined number of simulated days. It can be seen that, on the one hand, the simulator can show the results from different perspectives, which include not only the accumulated optimization objective values over the period of running it, but also the results that cannot be known by the optimization algorithm, such as the number of defects, the number of unsatisfied trips. On the other hand, these results are used for the final evaluation which is based on the "real" results and not on the raw optimization results. The raw optimization results cannot be used as actual results due to the reason that the optimizer does not have full knowledge of the future.

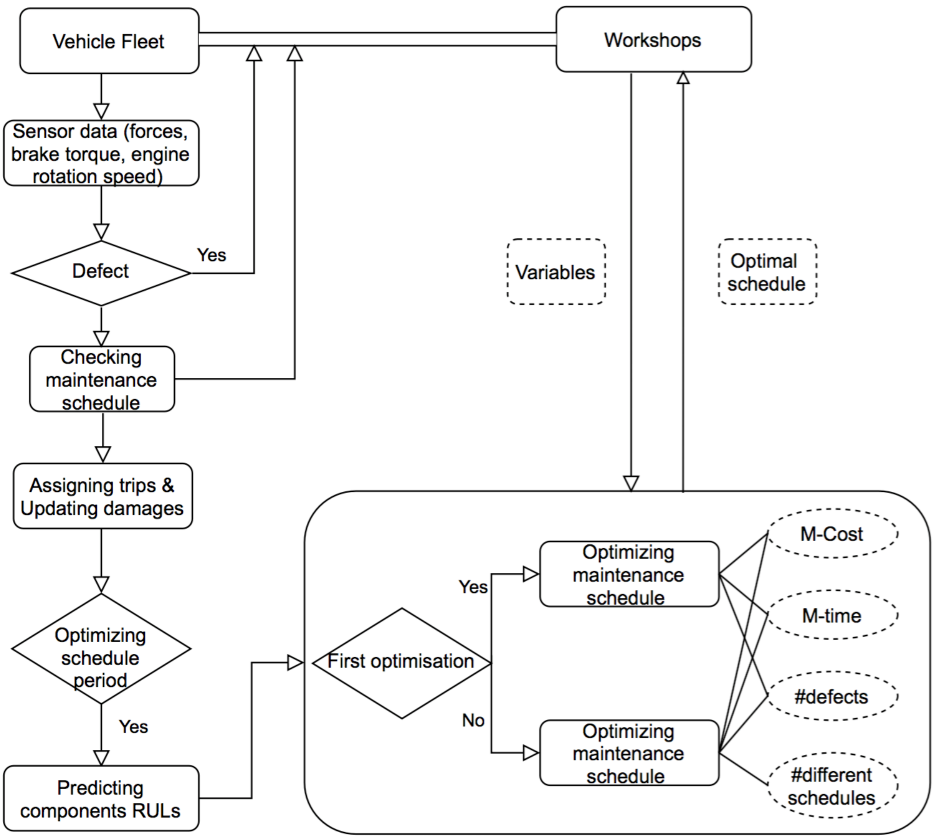


Figure 6.6: Daily workflow of the simulator.

Figure 6.6 shows the workflow of the simulator. The flow is executed on a daily basis. At the beginning of each day, vehicles in workshops are checked and sent back to work when their maintenance is done, meaning the damage of these components is set to zero. Next, the damage of each component is investigated and vehicles are sent to workshops when defects occur, which means the damage percentage of a component reaches 100%. In the case of a defect, the car is sent to a random workshop. Afterwards, the maintenance schedule is checked and the vehicles are sent to the assigned workshops if they are assigned to be maintained on that day. Hereafter, the driving trips of that day are assigned to the available cars and the damages of components are updated. Lastly, when it is the day to generate a new maintenance schedule, the RUL distributions of components are predicted, and the maintenance schedule is optimized. In the case of generating the first maintenance schedule, only three objectives are employed. Later on, the stability of the schedule is involved in the optimization procedure as an extra objective. After obtaining the PF approximation from each optimization, the knee point on the PF is picked and deployed as the new schedule to replace the current schedule to maintain the vehicle fleet.

6.2.4 Experiments

To show and observe the impact of different maintenance strategies clearly, the simulator runs under the scenarios with the following combinations of parameters:

- the simulation time: 700 days,
- the size of the vehicle fleet: 20 cars with 2 workshops, 20 cars with 5 workshops,
- the frequency of updating schedule: weekly, monthly,
- the computing budget of optimization: 100000, 500000,
- the optimization algorithm: basic MOEA, preference based MOEA, dynamic basic MOEA, dynamic preference based MOEA.

The results of the prediction-based optimization algorithms are also compared with fixed-interval maintenance scheduling. To set the fixed-interval maintenance, firstly the simulator is run without the maintenance schedule. In this case, each component breaks until its due date or its damage reaches 100%, then it is maintained and sent back to perform the driving tasks again. The average mileages are obtained for 13 components to be maintained, which include engine, 4 brake pads (front left, front right, rail left and rail right respectively), 4 tires and 4 springs. They are used as the condition for the maintenance in the fixed-interval maintenance scheduling approach, i.e., if a component reaches its corresponding average mileage, it is sent for maintenance.

Tables 6.3 - 6.5 show the results from the simulator. In these tables, the first column shows which algorithm has been applied in the maintenance period (700 days in the experiments). To optimize the maintenance schedule, four different optimization algorithms have been applied and compared. Basic and preference-based algorithms only take into account three objectives: cost, time and the number of failures. Dynamic basic and dynamic preference-based algorithms involve the fourth objective (the stability of the schedule). It means that basic and preference based algorithms handle multi-objective optimization problems, and dynamic basic and dynamic preference based algorithms deal with many-objective optimization problems [84]. Many-objective optimization focuses on solving optimization problems with four or more objectives and it forms a special and important case of multi-objective optimization problems. Solving many-objective optimization problem is more challenging for MOEAs due to the high computational cost resulting from increased evaluation of the number of points required for the PF approximation.

The other columns in these tables give the final results according to the simulation. These results include the number of failures (#defects), the total cost (cost), the total maintenance time (time), the number of changed schedules (#ch-sch), the number of unsatisfied trips (#un-trips) and the number of scheduled maintenance activities (#sch-act). Since the maintenance schedule is based on the average mileage and is not updated for the fixed-interval maintenance, the number of changed schedules is not applicable in this case. The parameter setting for each scenario has also been given in the table, for example, "schedule-update: monthly; #evaluations: 100000" refers to the scenario when the maintenance schedule is updated monthly and the computing budget of the optimization algorithm is 100000. All experimental data are the average results from five runs, in each run a different seed for the simulation is used.

Table 6.3 shows the experimental results from two different scales of the problem: one is 20 cars and 2 workshops; another is 20 cars and 5 workshops. When there are more workshops, the maintenance time can get reduced because there is less chance for vehicles to wait for their maintenance. This results in a decrease of the number of unsatisfied trips because the waiting time in workshops is now used to execute trips. Accordingly, the number of maintenance jobs (both the number of scheduled maintenance activities and the number of defects) increases. So does the maintenance cost. When comparing the results from these two problems, it can be seen that the data match this logic.

When comparing the results of dynamic algorithms with four objectives and their corresponding algorithms without the fourth objective, it can be seen that dynamic algorithms can always reduce the number of changed schedules, but this also means they have to sacrifice the other objectives to some extent. In some industrial scenarios, the stability objective plays a critical role. For example, in the case of aircraft maintenance, some maintenance activities are conducted during the intervals between takeoffs and landings, the change on the maintenance schedule may make an impact on the schedule of this flight and also might disrupt other flights, a rescheduling typically causes significant communication costs.

Next, with more computing budget for the optimization algorithms (i.e., the number of objective function evaluations is 500000.), it can be seen that the overall results after running the simulator get improved for three objective optimization (i.e., for basic and preference based algorithms.). The results here refer to the objectives that the algorithms optimize. However, for the dynamic algorithm, the results with more computing budget are sometimes mutually dominated with the results from using a smaller computing budget. For example, the number of defects can be reduced with the larger computing budget, but the total maintenance cost cannot get improved by more computing budget. This is led by the complexity of many-objective optimization. When determining the schedule to be deployed from the PF, the knee point is chosen. However, in four dimensional space, a small variation can lead to a big impact on the final result, especially on the accumulated results of multiple optimizations. With monthly schedule updates, the optimization algorithm is executed 22 times during one simulation run of 700 days.

When the schedule is updated more often, i.e., weekly, a reduction of the defect number is observed. Apparently, updating the maintenance schedule more frequently can promote the accuracy of it because the predicted RUL is more accurate. At the same time, an improvement of the total cost can be seen. The reason for the reduction of the total cost also comes from the accuracy of the schedule and the resulting decrease of the penalty cost which arises when the vehicle is maintained before it is broken, i.e., the cost for too-early maintenance. When updating the maintenance schedule more often, the maintenance time can not always get improved because the number of maintenance tasks does not always get decreased, the maintenance tasks may increase due to the accuracy of the schedule and the resulting increase on the number driving tasks which have been executed.

When comparing the preference based algorithm and basic algorithm, for both three objective and four objective optimization, it can be seen that the results of the preference based algorithm are usually better than its corresponding basic algorithm for the scenario of five workshops. However, if there are only two workshops, the waiting of vehicles for their maintenance results in a performance degradation of the preference based algorithms. The similar working tasks of the vehicles lead to the phenomenon that the scheduled maintenance times for some vehicles are close, and this leads to that the workshops run out of capacity sometimes but become idle at other times. Therefore, a good solution would be to offer more workshops for the fleet, at the same time, these workshops can also work for other tasks besides for the fleet.

Lastly, when comparing with the fixed-interval maintenance, there are more defects, maintenance time and unsatisfied trips for the fixed-interval maintenance. Since most maintenance tasks are caused by defects, the too-early maintenance cost drops dramatically and this leads to the decrease of the total cost.

Table 6.3: Optimization results of different maintenance scenarios over 5 runs.

	20 cars & 2 workshops					
Algorithm	#defects	cost	time	#ch-sch	#un-trips	#sch-act
Fixed-interval	226	474965	6269	NA	212450	52
	schedule-update: monthly; #evaluations: 100000;					
Basic	46	680666	4282	4509	121000	148
Preference	50	690871	4179	4630	112150	150
Dynamic basic	73	676149	5510	4023	175800	154
Dynamic preference	66	688934	4732	3729	137200	159
	schedule-update: monthly; #evaluations: 500000;					
Basic	39	675374	3936	4553	101750	150
Preference	40	677331	3903	4526	101950	145
Dynamic basic	68	717046	5262	3777	161300	157
Dynamic preference	42	690131	4669	3240	140750	150
	schedule-update: weekly; #evaluations: 100000;					
Basic	32	624078	4565	22884	126200	166
Preference	35	646117	4103	23016	109700	168
Dynamic basic	67	633854	5758	19660	185450	150
Dynamic preference	50	628228	4626	18049	140200	161
	20 cars & 5 workshops					
Algorithm	#defects	cost	time	#ch-sch	#un-trips	#sch-act
Fixed-interval	330	747104	2996	NA	72750	92
	schedule-update: monthly; #evaluations: 100000;					
Basic	68	785777	1852	4877	27950	192
Preference	67	748044	1837	5012	25750	184
Dynamic basic	137	849203	2942	4466	62700	218
Dynamic preference	93	789592	2331	4247	42000	217
	schedule-update: monthly; #evaluations: 500000;					
Basic	55	756278	1725	4901	24850	182
Preference	50	718176	1649	4924	22550	177
Dynamic basic	125	831775	2754	4442	56950	223
Dynamic preference	91	797258	2130	4014	34750	210
	schedule-update: weekly; #evaluations: 100000;					
Basic	60	695181	1995	23973	35550	206
Preference	56	690720	1951	23982	31250	205
Dynamic basic	114	768697	3122	21715	77950	217
Dynamic preference	91	721193	2296	20397	44100	227

Besides the parameters which change the experimental environment, the variables in the optimization algorithm can also be adjusted to emphasize some aspects of the results. To reduce the number of failures of vehicles, the interval of the execution window is switched from $[\mu - 2\times\sigma,\ \mu + 2\times\sigma]$ to $[\mu - 3\times\sigma,\ \mu + \sigma]$. Table 6.4 shows the results of 20 vehicles and 5 workshops. After shifting the execution window forward, the dramatic drop of the number of defects is achieved and the descent rate reaches 83.21% on average. Simultaneously, this activates the rise of the maintenance cost.

Table 6.4: Adjust execution window to reduce the number of defects.

	20 cars & 5 workshops					
Algorithm	#defects	cost	time	#ch-sch	#un-trips	#sch-act
	schedule-update: monthly; #evaluations: 100000;					
Basic	7	886747	1872	4787	22050	227
Preference	10	840907	1834	4767	21450	223
Dynamic basic	28	1152867	3276	4143	63925	303
Dynamic preference	20	1032917	2399	4102	34350	285
	schedule-update: monthly; #evaluations: 500000;					
Basic	5	823662	1599	4826	15050	201
Preference	5	822518	1534	4804	12850	190
Dynamic basic	27	1133032	3618	3960	79100	292
Dynamic preference	22	979069	2339	3686	34950	258
	schedule-update: weekly; #evaluations: 100000;					
Basic	4	823683	1945	23496	25750	248
Preference	4	805841	1815	23472	22100	225
Dynamic basic	22	1070067	3092	20856	62450	287
Dynamic preference	15	864731	2313	19701	42350	231

It is worth noting that the problems with 20 vehicles and 13 components for each vehicle are already large scale scheduling optimization problems in terms of the domain of flexible job shop scheduling optimization. Moreover, the MOVFMSO problem is more complex than FJSS because the MOVFMSO problem needs to assign not only the workshops and maintenance times (sequences) for the maintenance activities, but also the combination of components for each activity. To investigate how scalable the proposed approach is, the questions asked are whether the algorithms can be applied to even larger fleet and whether consistent results can be achieved when the fleet becomes significantly larger. To this end, the fleet size has been increased to 50 vehicles and 15 workshops are available, the components to be maintained retain the same. Table 6.5 shows the simulator results and it can be observed that these results are consistent with the results presented earlier.

From the experimental results, some major insights on how to design schedules with respect to the objectives can be concluded as follows.

- Providing additional workshops can help reduce the overall maintenance time.

Table 6.5: Increase Problem Size.

		50 cars & 15 workshops				
Algorithm	#defects	cost	time	#ch-sch	#un-trips	#sch-act
Fixed-interval	864	1878632	7118	NA	162850	209
	schedule-update: monthly; #evaluations: 100000;					
Basic	45	2429887	5440	11933	63600	697
Preference	43	2395079	5104	12171	54600	669
Dynamic basic	109	2887019	7571	11598	127200	812
Dynamic preference	71	2751571	6481	11302	89300	792
	schedule-update: monthly; #evaluations: 500000;					
Basic	35	2320261	5016	12036	63600	635
Preference	32	2254907	4551	12176	42950	626
Dynamic basic	105	2990837	8454	11210	164700	853
Dynamic preference	70	2677516	6392	10971	87850	753
	schedule-update: weekly; #evaluations: 100000;					
Basic	24	2149010	5267	58736	70350	659
Preference	26	2141242	5047	58929	69100	643
Dynamic basic	102	2750433	6908	56272	107500	782
Dynamic preference	61	2418850	5819	55485	83400	706

- Moving the execution window to the left (earlier time) or updating the schedule more often can both be used to reduce the number of defects.

- Without introducing stability as an additional objective, schedule tends to be disrupted by dynamic updates.

- Both the use of the preference based algorithms and increasing the computing budget have a positive impact on the overall quality. However, the best way to improve the overall quality of the final results is to increase the number or capacity of workshops in combination with the preference based algorithms.

- Comparing the fixed-interval maintenance vs. prediction-based scheduling optimization, it can be concluded that fixed-interval maintenance leads to an unsatisfactory performance in terms of number of defects, whereas prediction-based scheduling optimization finds a balanced trade-off satisfying all objectives to high extents. Therefore, the extra computational effort required to make predictions and perform optimizations is well justified.

- The results on the large-scale benchmark problem with 50 vehicles indicate that the proposed algorithms can also handle larger problems and the main conclusions, as summarized in the previous points, remain the same.

6.2.5 Summary and Outlook

Since optimization algorithms are required to regularly update maintenance schedule in a dynamic environment, the proposed multi-objective evolutionary algorithms are extended to dynamic many-objective evolutionary algorithms that take stability as the fourth objective to aim for the robustness of maintenance schedule. The vehicle fleet maintenance scheduling optimization simulator has been developed, which can be used as a scalable benchmark for optimizing vehicle fleet maintenance schedules in an industrially relevant setting. The simulator and benchmark problems have been inspired by the instances faced by a taxi company with up to 50 cars. The proposed MOEAs can be compared and tested easily in the simulator in a rolling-horizon fashion. Parameters and algorithms can be adjusted to imitate various scenarios. Therefore, although the implementation of the approach is demonstrated in the example of taxi fleets, the proposed approach can be adapted to different industrial applications, for example, the maintenance of trucks, vessels, aircraft, etc.

The size of problems in the experiments is up to 50 vehicles and 13 components for each vehicle. Still, one might imagine the problems of even larger scale, and finding out the limit of the fleet size that the algorithm can handle would be an interesting future research. However, for this, high performance computing environments and parallel computing might be required, especially when it comes to statistical studies. In this work, to maintain clarity of presentation the dynamically changing element is so-far restricted, but in the future work additional dynamic elements and uncertainties should be considered. For example, the uncertainty on the maintenance duration could be modeled, as in [52], the presence of cost uncertainty in [31], etc.