A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling

Xiao-Ning Shen[a,*], Leandro L. Minku[b], Naresh Marturi[c], Yi-Nan Guo[d],Ying Han[a]

[a] *B-DAT, CICAEET, School of Information and Control, Nanjing University of Information Science and Technology, Nanjing 210044, China*

[b] *Department of Informatics, University of Leicester, University Road, Leicester LE1 7RH, United Kingdom*

[c] *Extreme Robotics Lab, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom*

[d] *School of information and control engineering, China university of Mining and Technology, Xuzhou 221116, China*

Abstract Software project scheduling is the problem of allocating employees to tasks in a software project. Due to the large scale of current software projects, many studies have investigated the use of optimization algorithms to find good software project schedules. However, despite the importance of human factors to the success of software projects, existing work has considered only a limited number of human properties when formulating software project scheduling as an optimization problem. Moreover, the changing environments of software companies mean that software project scheduling is a dynamic optimization problem. However, there is a lack of effective dynamic scheduling approaches to solve this problem. This work proposes a more realistic mathematical model for the dynamic software project scheduling problem. This model considers that skill proficiency can improve over time and, different from previous work, it considers that such improvement is affected by the employees' properties of motivation and learning ability, and by the skill difficulty. It also defines the objective of employees' satisfaction with the allocation. It is considered together with the objectives of project duration, cost, robustness and stability under a variety of practical constraints. To adapt schedules to the dynamically changing software project environments, a multi-objective two-archive memetic algorithm based on Q-learning (MOTAMAQ) is proposed to solve the problem in a proactive-rescheduling way. Different from previous work, MOTAMAQ learns the most appropriate global and local search methods to be used for different software project environment states by using Q-learning. Extensive experiments on 18 dynamic benchmark instances and 3 instances derived from real-world software projects were performed. A comparison with seven other meta-heuristic algorithms shows that the strategies used by our novel approach are very effective in improving its convergence performance in dynamic environments, while maintaining a good distribution and spread of solutions. The Q-learning-based learning mechanism can choose appropriate search operators for the different scheduling environments. We also show how different trade-offs among the five

objectives can provide software managers with a deeper insight into various compromises among many objectives, and enabling them to make informed decisions.

Keywords Metaheuristics; Dynamic software project scheduling; Multi-objective memetic algorithms; Mathematical modeling; Q-learning

1 Introduction

With the rapid development of the software industry, software companies are confronted with a highly competitive market environment. In order to win the market, as well as to meet project requirements (deadlines, budget, etc.), efficient and effective software project schedules need to be adopted [29, 36]. Nevertheless, it is not uncommon for software companies to face software project failure due to inappropriate project schedules. For instance, it is reported that more than 40% of the software projects in China are unsuccessful due to incoherent scheduling of tasks and human resources [4]. Another example is NASA's checkout launch control software project, which involved more than 400 people and had to be cancelled after having gone over budget [21].

The software project scheduling problem (SPSP) consists in allocating employees to tasks over a project timeline, so that the required objectives (project cost, duration, etc.) can be achieved subject to various constraints. This problem is particularly complex and challenging for large-scale software projects [27], which may involve dozens or even hundreds of software engineers cooperating to complete a large number of tasks. This is because the size of the search space of potential allocations of employees to tasks is extremely large for such projects. Scheduling large software projects manually can be very time-consuming and result in inefficient and unsatisfactory schedules. Given that software projects have become increasingly large, it is desirable to have methods to help project managers with the SPSP.

With the development of search-based software engineering (SBSE) [18], many researchers have formulated SPSP as a search-based optimization problem, and employed evolutionary algorithms (EAs) [13, 15] to search the large decision space and provide near-optimal schedules, automating the task finding good allocations and helping the software manager to make a final decision.

Real-world software projects often suffer working environment changes due to unpredictable events such as employees' leave of absence, change of requirements, addition of high-priority tasks, etc. Moreover, the project activities are often subject to uncertainties. For example, the amount of effort required to develop tasks may have been underestimated or overestimated, and the number of resources required for completing a task may change due to variations of the task specification [31]. In such cases, an optimal schedule produced by a static scheduling method may get severely deteriorated during the project execution. It is thus of great importance to employ dynamic scheduling techniques to adapt schedules for dynamically changing environments, as well as to reduce the impact of uncertainties.

So far in the literature, very few studies have dealt with dynamic software project scheduling in uncertain project environments. Proactive scheduling was used for software projects with uncertainties [5, 16, 17, 20, 22], and dynamic resource rescheduling was designed to react to new project arrivals [43]. Our previous study [35] was the first research work dealing with the mathematical modeling and dynamic

scheduling of the multi-objective dynamic software project scheduling problem (MODSPSP). This previous work addresses both uncertainties and dynamic events occurring during a project lifetime.

Although a few interesting results have been reported by the above studies, difficulties still exist and need to be further addressed. One of the difficulties is that there is a gap between the constructed SPSP mathematical formulation and real-world software projects. Since human resources are one of the main project resources, software project managers have paid much attention to the influence of employees' subjective properties on project success. However, only some basic properties of employees and tasks are considered by existing SPSP models [1, 2, 4, 27, 35], such as the specific skills possessed by each employee. In order to formulate a more practical model, more properties of human resources and tasks should be taken into account, as well as their possible variations. For example, a certain employee may have a strong willingness to take part in a specific task, and his/her skills proficiency can improve with experience. Besides, employees' satisfaction for the allocation is an important criterion that a software manager often considers when scheduling, but it has never been regarded as an objective to be optimized in the existing models.

Another difficulty is the lack of effective dynamic scheduling approaches for solving MODSPSP. In MODSPSP, the search space is large, and the project environment varies dynamically. However, the current search-based dynamic scheduling approaches for this problem find solutions based on fixed search operators, and lack self-learning mechanisms that could exploit the environment features to guide the search direction. Therefore, existing approaches have weak ability to adapt and may suffer from slow convergence speed and/or premature convergence. It is therefore hard for them to find optimal schedules in dynamically changing environments [24]. In particular, different environment conditions may require different search operators in order to find good solutions more efficiently and effectively. Existing approaches for MODSPSP are unable to deal with that.

With the aim of covering the shortage of existing methods, this paper makes two main contributions: (i) It proposes a more realistic model of MODSPSP, highlighting the influence of employees' subjective properties on project success. Different from previous work, our model considers several additional employee properties, namely the fact that their skill proficiency can be improved over time and that such improvement is affected by their motivation and learning ability and the skill's difficulty. Moreover, it takes into account the impact of employees' satisfaction by considering it as an extra objective in addition to project duration, cost, robustness and stability. (ii) It proposes a multi-objective two-archive memetic algorithm based on Q-learning (MOTAMAQ) to solve the formulated MODSPSP in a proactive-rescheduling way. The key idea of the approach is to learn the appropriate global and local search methods of MA adaptively in dynamically changing software project environments through Q-learning.

To validate the effectiveness of our approach, 18 dynamic SPSP benchmark instances and 3 instances derived from real-world software projects were used in our experimental studies. Our results indicate that the introduction of self-adaptive learning mechanism based on Q-learning helps to improve the convergence performance of MOTAMAQ. By cooperating with the global search operators, the problem-specific local search operators enhance the local search ability of the algorithm. Besides, the

maintenance of two archives that promote convergence and diversity separately can deal with the 5 objectives in MODSPSP effectively.

The remainder of this paper is organized as follows. Section 2 gives background information and an overview of related work. Our proposed MODSPSP mathematical model is presented in Section 3. In Section 4, our Q-learning-based proactive-rescheduling framework is presented, and the proposed rescheduling approach MOTAMAQ is introduced. Section 5 presents the experimental studies. Conclusions are drawn in Section 6.

2 Background and related work

2.1 Memetic algorithms and Q-learning

Memetic algorithm (MA) is a meta-heuristic approach that mimics the process of culture evolution, and imitates the mutation process supported by specialized knowledge through local heuristic search [28]. MA combines population-based global search with individual-based local search. Compared to traditional EAs, MAs can find high quality solutions more efficiently in many applications because local search abilities are enhanced [30]. Some multi-objective memetic algorithms have been used for solving multi-objective optimization problems [12, 25, 45]. However, environments change dynamically in MODSPSP. Therefore, MA is likely to exhibit distinct search performance if different global and local search operators are adopted for different environment states. Thus, if the algorithm can capture the state of the current environment, and learn the current best evolutionary operators by itself, the search efficiency of MA for solving MODSPSP will be greatly improved.

Reinforcement Learning (RL) is an effective method to learn an optimal behavior by trial and error interactions between an agent and a dynamic environment [37, 41]. The agent can perceive the environment, select an action to execute in the current state, and then the environment feedbacks a reward or penalty signal to the action taken. The interactions iterate until the agent learns the action decision-making policy that maximizes the accumulated reward. The above process makes the learning strategy of RL have long-term effects. Q-learning [6, 19] is one of the famous RL approaches. Its goal is for an agent to learn the optimal long-term expected reward value Q(s, a) for each pair of state (s) and action (a). If the fitness value of an individual is regarded as a reward, then there is a high conceptual and structural compatibility between Q-learning and MA [32]. Based on the above analysis, if a self-learning multi-objective memetic algorithm is designed by combining both the merits of MA and Q-learning, it may have a great potential to solve complex dynamic multi-objective problems like MODSPSP.

2.2 SPSP task-based models

The task-based model is a very popular formulation of the SPSP in the search-based software engineering literature [27]. In this model, there are a group of employees and a set of tasks. Each employee has the properties of monthly salary, a set of skills, etc. An employee can perform several tasks concurrently during a working day. Each task also has some properties such as task effort and a set of required skills. Execution of the tasks should follow a task precedence graph (TPG), which provides information about the tasks that need to be completed before commencing new ones. The tasks and TPG are considered as the project to be scheduled. SPSP consists in determining which employees are allocated

to each task, and specifying the dedication of each employee to the assigned tasks, with the objectives of minimizing project cost, duration and so on. Constraints of no overwork, required task skills, etc, are also considered [1].

Many researchers have investigated task-based models in static environments. Chang et al. [2] proposed a task-based model in which overwork was regarded as an extra objective in addition to project duration and cost. Alba and Chicano [1] and Minku et al. [27] have employed similar models to the one specified in [2], where the total dedications of each employee to all active tasks were not allowed to exceed a predefined maximum value specified by the company rules. However, in both these works, overwork was treated as a constraint. Crawford et al. [7] established a construction graph according to the task-based model to adapt a Max-Min Ant system to solve SPSP. Wu et al. [42] proposed an evolutionary hyper-heuristic to solve the static SPSP, which adaptively chose both crossover and mutation operators during the search. Chicano et al. [5] and Luna et al. [24] formulated the SPSP as a multi-objective optimization problem where duration and cost were optimized simultaneously based on Pareto dominance. To make the allocation more flexible, a task-based model with an event-based scheduler was proposed by Chen and Zhang [4]. In this work, an original schedule was adjusted at events. To make the task-based model more practical, a time-line model that divided the task duration into various time slices was developed by Chang et al. [3]. Employees were allocated to tasks in discrete time slices iteratively, and the accumulated fitness values were evaluated for each solution. In this manner, more features such as re-assignment of employees, different skill proficiencies, distinct salaries for normal work and overtime work could be introduced. Nevertheless, too many subjective parameters need to be set and tuned to use this model, which could make it difficult for software managers to use. Besides, sensitivity of the algorithm's performance to various parameters was unknown [27], and a large amount of system instability would be induced because the tasks were scheduled separately within different time slices [4].

Considering that it is extremely challenging to obtain all the accurate information beforehand, in our previous work [35], we formulated software project scheduling as a dynamic scheduling problem with task effort uncertainty and three types of dynamic events (employee leaves, employee returns and new task arrivals). Then, we proposed a dynamic version of the task-based model, where both the efficiency related objectives (project duration and cost) and the objectives concerning dynamic environment (robustness and stability) are considered together.

In most of the above-mentioned models, only some basic properties of employees and tasks are considered. Properties such as the ability and motivation of employees to improve their skills proficiency, and satisfaction with the generated schedule are not taken into account. This makes the existing models inconsistent with real cases.

2.3 Software project scheduling approaches for dynamic environments

A few studies have proposed approaches to deal with uncertainties in software projects. Most of them adopted proactive scheduling. Hapke et al. [17] translated a fuzzy scheduling problem into a set of deterministic problems by describing the uncertain activity parameters through L-R fuzzy numbers. To reduce the effect of uncertainties on project performance, the most appropriate remedial action was chosen according to the project goal using the simulation-based method provided by Lazarova-Molnar and Mizouni [22]. Gueorguiev et al. [16] adopted an MOEA to search the trade-off solutions between completion time and robustness to new tasks. Similarly, Chicano et al. [5] also used an MOEA to solve the

multi-objective SPSP that considered both the employee productivity in performing different tasks and robustness to the inaccuracies of task effort estimations. Harman [18] performed a number of Monte Carlo simulations according to a baseline schedule and an event list, so that a schedule under uncertainties was generated. Vasant et al. [39] considered a project with imprecise activity times and proposed ranking methods based on fuzzy mathematical techniques.

In the existing literature, very few works have studied resource rescheduling in response to the dynamic events occurring during software development. Xiao et al. [43] considered one type of disruptive event, which stood for the addition of a new project, and implemented the rescheduling of three new project arrivals. To address both the uncertainties and dynamic events, in our previous work [35], we have proposed a proactive-rescheduling approach based on ε-MOEA [9] to solve the dynamic software project scheduling problem. A robust schedule is produced predictively with regard to the project uncertainties, and then the previous schedule is revised by the rescheduling approach in response to critical dynamic events. However, with our previous approach, the operators in the dynamic scheduling approach are fixed, and it is not possible to select appropriate search operators adaptively according to the current project state.

To resolve this problem, in this paper, a more practical dynamic version of the task-based mathematical model for MODSPSP is formulated, which addresses the effect of several human factors on project success. For instance, the model considers improvement of employees' skill proficiency over time, and includes an objective to take employees' satisfaction into account, together with project duration, cost, robustness and stability. A multi-objective two-archive memetic algorithm is proposed to solve the formulated MODSPSP in a proactive-rescheduling way, which can learn the appropriate global and local search operators of the memetic algorithm adaptively in the dynamically changing project environment based on Q-learning.

3 MODSPSP problem formulation and mathematical modelling

The MODSPSP mathematical model constructed in this paper is an improvement of the one presented in our previous work [35]. Different from that work, it considers that the skill proficiency of each employee can be improved over time. It also considers the learning ability and the motivation of each employee to learn new knowledge, and the difficulty of each skill. Besides, the degree with which each employee is willing to engage with each skill (and thus in each task) is considered, leading to a new objective to represent employees' satisfaction. This objective is considered in addition to the objectives of project duration, cost, robustness and stability considered in previous work [35].

3.1 The proactive-rescheduling mode

To address the dynamic features in software project scheduling, one type of uncertainty (the task effort uncertainty), and three dynamic events (new task arrivals, employee leaves and employee returns) are taken into account. Among them, urgent task arrivals, employee leaves and returns are considered as critical events, and regular task arrivals as trivial events.

A proactive-rescheduling mode is employed. Initially, a robust schedule for all tasks and employees at the initial time t_0 is produced by a proactive scheduling approach, considering the objectives of project efficiency (duration and cost), schedule robustness (sensitivity of the schedule to task effort uncertainty),

and employees' satisfaction (willingness of the employees to be allocated to tasks) together. To reduce the rescheduling frequency, a critical-event-driven mode is adopted. Once a critical event occurs, the rescheduling approach is triggered, optimizing the objectives of project efficiency, schedule robustness, system stability (deviations between the new and original schedules) and employees' satisfaction, simultaneously. Trivial events (regular task arrivals) are not scheduled until the next critical event takes place. However, if the new regular task has to start before the occurrence of next critical event based on the TPG, a heuristic method is used to allocate it [35]. The time when a new schedule is regenerated is called the *rescheduling point*, and is denoted as t_l ($l=1,2,\ldots$). The unit of t_l is month. Newly generated schedule is executed in the project until the next critical event occurred, at which time the above rescheduling approach is driven again. This process lasts until the entire project is finished.

3.2 Skills' properties

Assume that the project to be scheduled requires in total S skills. The degree of difficulty SD^k is attributed to the kth skill ($k=1,2,\ldots,S$). A greater value of SD^k indicates that the kth skill is more difficult to be grasped.

3.3 Employees' properties

Table 1 describes employees' properties. Assume that M employees could get involved in a project. The skills that an employee e_i possesses at the rescheduling point t_l are denoted by $e_i^{skills}(t_l)=\{pro_i^1(t_l),pro_i^2(t_l),\ldots,pro_i^S(t_l)\}$, where $i=1,2,\ldots,M$ and $pro_i^k(t_l)\in[0,\mathrm{C}]$ ($k=1,2,\ldots,S$) is a fractional score which measures the proficiency of e_i for the k^{th} skill at t_l. If $pro_i^k(t_l)=0$, then e_i does not have the k^{th} skill, and if $pro_i^k(t_l)=\mathrm{C}$, then e_i totally masters the k^{th} skill. To describe the proficiency in a more detailed way than in [35], C is set to be 100 here.

According to interviews with software managers, $pro_i^k(t)$ is assumed to be improved with the time t as follows:

$$pro_i^k(t)=\tanh\left(a_i^k\left(t-t_0+I_i^k\right)\right)\cdot\mathrm{C}=\frac{e^{a_i^k\left(t-t_0+I_i^k\right)}-e^{-a_i^k\left(t-t_0+I_i^k\right)}}{e^{a_i^k\left(t-t_0+I_i^k\right)}+e^{-a_i^k\left(t-t_0+I_i^k\right)}}\cdot\mathrm{C},\quad t\geq t_0 \tag{1}$$

where, *tanh*(.) is the hyperbolic tangent function, the parameter a_i^k describes the proficiency growth rate, t_0 is the initial scheduling time, and I_i^k is derived from the initial proficiency $pro_i^k(t_0)$ of the employee. More specifically, $a_i^k=\frac{e_i^{LA}\cdot e_i^{MO}}{SD^k}$, where e_i^{LA} represents the learning ability factor of employee e_i. Greater values of e_i^{LA} indicate better learning ability of e_i. The learning ability includes the ability of

logic thinking, reasoning, understanding, data analysis, generation, abstraction, etc. The factor e_i^{MO} is the motivation factor of employee e_i, which measures the internal drive of e_i to improve skill proficiency. Motivation may be influenced by various aspects, such as individual interests, working attitude, habit formed through a long period of time, salary, sense of worthiness, family environment, company's culture, etc. A greater value of e_i^{MO} indicates a stronger general motivation of e_i to master skills. In a software project team, different employees have different learning abilities and motivations, and for a mature employee, these two characteristics are relatively stable. The learning ability factor of an employee can be scored by the software manager and fellow team mates based on his/her performances in the previous projects. Meanwhile, from interviews with real-world software managers, the motivation factor can be measured by a specialized questionnaire, which has been adopted in many companies. Greater values of e_i^{LA} and e_i^{MO}, and smaller values of SD^k, lead to greater values of a_i^k, which means a faster improvement of the proficiency. The parameter a_i^k can also be related to other factors, e.g. specialization. This will be further studied in our future work.

Given the initial proficiency $pro_i^k(t_0)$ at the beginning of the project, the value of I_i^k in (1) can be obtained using $I_i^k = \frac{\text{atanh}(\frac{pro_i^k(t_0)}{\text{C}})}{a_i^k}$, which indicates the time span it takes for an employee e_i to achieve the initial proficiency $pro_i^k(t_0)$, and *atanh*(.) is the inverse hyperbolic tangent function. It is worth noting that I_i^k is a fixed parameter once $pro_i^k(t_0)$ and a_i^k are given. The changing curve of $pro_i^k(t)$ with time t is illustrated in Fig.1.

The degree with which each employee is willing to engage with each skill is denoted by $e_i^{ED} = \{ED_i^1, ED_i^2, \ldots, ED_i^S\}$, where $ED_i^k \in [0,1]$ $(k = 1,2,\ldots,S)$ is a fractional score which measures the degree of e_i for the kth skill. $ED_i^k = 0$ means that e_i is not willing to engage with the kth skill at all, and $ED_i^k = 1$ means that e_i is willing to engage with the kth skill fully. This property will be used to compute one of the objectives of MODSPSP (see section 3.5).

We use $e_{ij}^{Proficiency}(t_l) = \prod_{k \in req_j \cap skill_i} \frac{pro_i^k(t_l)}{\text{C}}$ to indicate the proficiency of e_i for task T_j at t_l, where req_j is the set of specific skills required by task T_j, and $e_{ij}^{Proficiency}(t_l) \in [0, 1]$.

Each employee e_i $(i = 1,2,\ldots,M)$ also has a maximum dedication e_i^{maxded} (maximum percentage of

his/her time that can be spent on the project), a salary paid for normal working hours $e_i^{norm\_salary}$, and a salary paid for overwork hours $e_i^{over\_salary}$.

During the execution of the project, e_i may leave, and return later. $e_i^{available}(t_l)$ is used to indicate the availability of e_i. We use $e\_ava\_set(t_l)$ to denote the set of available employees at t_l, i.e., $e\_ava\_set(t_l)=\{e_i \mid e_i^{available}(t_l)=1,\ i=1,2,\ldots,M\}$.

In summary, each employee has some time-invariant properties (e_i^{LA}, e_i^{MO}, e_i^{ED}, e_i^{maxded}, $e_i^{norm\_salary}$, $e_i^{over\_salary}$), and also some time-related properties ($e_i^{skills}(t_l)$, $e_i^{available}(t_l)$, $e_{ij}^{Proficiency}(t_l)$). Since $pro_i^k(t_l)$ can be improved over time, $e_{ij}^{Proficiency}(t_l)$, which means the proficiency of e_i for task T_j at t_l, is also time-variant.

It is worth noting that although the skill level improvement has been considered in [3], the model just took the training hours into account. In contrast, our model relates the skill proficiency growth rate to both the human factors (motivation, learning ability) and the skill difficulties, which is more appropriate for the real cases. Meanwhile, the "learning" mentioned here not only indicates the training course used in [3], but also includes other ways of study such as self-learning.

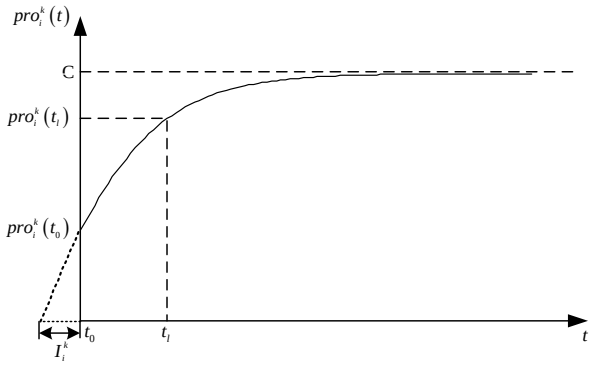

Fig. 1. Illustration of the changing curve of $pro_i^k(t)$ with time *t*.

Table 1

Properties of each employee.

name	description
e_i^{LA}	The learning ability factor of employee e_i. Greater values of e_i^{LA} indicate better learning ability of e_i.
e_i^{MO}	The motivation of employee e_i. A greater value of e_i^{MO} indicates a greater motivation of e_i to improve skill proficiency.
e_i^{ED}	The degree with which each employee is willing to engage with each given skill is denoted by $e_i^{ED}=\{ED_i^1,ED_i^2,\ldots,ED_i^S\}$, where $ED_i^k \in [0,1]$ ($k=1,2,\ldots,S$) is a fractional score which measures the degree of e_i for the kth skill. $ED_i^k=0$ means that e_i hopes not to have to engage with the kth skill at all, and $ED_i^k=1$ means that e_i is willing to engage with the kth skill fully.
e_i^{maxded}	The maximum dedication of e_i to the project, which means the percentage of a full-time job e_i is able to work. $e_i^{maxded}=1$ means e_i can dedicate all the normal working hours of a month to the project. Part-time jobs or overtime working are allowed by setting e_i^{maxded} to a value smaller or bigger than 1, respectively. For example, $e_i^{maxded}=1.2$ indicates e_i is allowed to work up to 120% of the normal working time.
$e_i^{norm\_salary}$	The monthly salary of e_i for his/her normal working time.
$e_i^{over\_salary}$	The monthly salary of e_i for his/her overtime working time.
$e_i^{skills}(t_l)$	The skill indicator set of employee e_i at t_l ($l=0,1,2,\ldots$), $e_i^{skills}(t_l)=\{pro_i^1(t_l),pro_i^2(t_l),\ldots,pro_i^S(t_l)\}$.
$skill_i$	The set of specific skills possessed by e_i. It can be converted from $e_i^{skills}(t_l)$, where $skill_i=\{k \mid pro_i^k(t_l)>0,\ k=1,2,\ldots,S\}$.
$e_i^{available}(t_l)$	A binary variable which indicates whether e_i is available or not at t_l. $e_i^{available}(t_l)=1$ means e_i is available at t_l, and $e_i^{available}(t_l)=0$ shows e_i is unavailable at t_l.
$e_{ij}^{Proficiency}(t_l)$	The proficiency of e_i for task T_j at t_l. $e_{ij}^{Proficiency}(t_l)=\prod_{k\in req_j \cap skill_i} \frac{pro_i^k(t_l)}{C}$ (req_j is the set of specific skills required by task T_j), and $e_{ij}^{Proficiency}(t_l) \in [0,\ 1]$.

3.4 Tasks' properties

By t_l, assume $(N_I+N_{new}(t_l))$ tasks have been regarded as part of the project in total, among which N_I tasks existed at the initial time of the project, and $N_{new}(t_l)$ new tasks were released in the project one-by-one during the time span $(t_0,t_l]$. Properties of each task T_j ($j=1,2,\ldots,N_I+N_{new}(t_l)$) are described in Table 2, where T_j^{skills}, req_j, $T_j^{est\_tot\_eff}$ are considered as time-invariant, and $T_j^{unfinished}(t_l)$, $T_j^{available}(t_l)$, TPG are time-related. It is worth noting that, even though $T_j^{est\_tot\_eff}$ is considered time-invariant, it may involve uncertainty. For example, the estimated effort may be inaccurate.

At t_l, it is possible that the task T_j has finished (marked by $T_j^{unfinished}(t_l)$), or T_j is unfinished but

unavailable (marked by $T_j^{available}(t_l)$) because it cannot be implemented temporally due to one or several employees' leaves resulting in at least one of the skills required by T_j not being grasped by any of the remaining employees. The set $T\_ava\_set(t_l)$ denotes the set of available tasks at t_l, i.e., $T\_ava\_set(t_l) = \left\{T_j \mid T_j^{available}(t_l) = 1, j = 1,2,\ldots,N_1+N_{new}(t_l)\right\}$.

TPG is updated at each rescheduling point t_l, considering the cases of completion of a task, new regular task arrival, or new urgent task arrival. One can refer to our previous work [35] for the details of TPG update.

Table 2

Properties of each task.

name	description
T_j^{skills}	The skill indicator set of task T_j. $T_j^{skills} = \left\{sk_j^1, sk_j^2, \ldots, sk_j^S\right\}$, where $sk_j^k = 1$ ($k = 1,2,\ldots,S$) indicates the k^{th} skill is required by T_j, and $sk_j^k = 0$ means not.
req_j	The set of specific skills required by T_j. It can be converted from T_j^{skills}, where $req_j = \{k \mid sk_j^k = 1, k = 1,2,\ldots,S\}$.
$T_j^{est\_tot\_eff}$	The initially estimated effort required to complete task T_j in person-months. The task effort uncertainty of T_j is assumed to follow a normal distribution $N(\mu_j, \sigma_j)$, where μ_j and σ_j are the mean and standard deviation, respectively. Here, we set $\mu_j = T_j^{est\_tot\_eff}$.
$T_j^{unfinished}(t_l)$	A binary variable indicating whether T_j has finished by t_l. $T_j^{unfinished}(t_l) = 1$ means that T_j is unfinished at t_l, and $T_j^{unfinished}(t_l) = 0$ means that T_j has finished by t_l.
TPG	An acyclic directed graph with tasks as nodes and task precedence as edges. TPG must be updated when a task finishes or a new task is added into the project. Here, $G(V(t_l), A(t_l))$ is used to represent the TPG at t_l, where $V(t_l)$ is the vertex set which includes all the arrived and unfinished tasks at t_l, i.e., $V(t_l) = \left\{T_j \mid T_j^{unfinished}(t_l) = 1, j = 1,2,\ldots,N_1+N_{new}(t_l)\right\}$, and $A(t_l)$ is the arc set which indicates the precedence relations among the tasks in $V(t_l)$.
$T_j^{available}(t_l)$	A binary variable indicating whether T_j is available or not at t_l. $T_j^{available}(t_l) = 1$ shows T_j is available at t_l, while $T_j^{available}(t_l) = 0$ means not. T_j is regarded as available at t_l if and only if the following three conditions are satisfied simultaneously: (1) T_j is unfinished at t_l, i.e., $T_j^{unfinished}(t_l) = 1$; (2) for any skill required by T_j, at least one of the available employees at t_l possesses the skill, i.e., if $k \in req_j$, then $\exists e_i$, s.t. $e_i \in e\_ava\_set(t_l) \wedge k \in skill_i$; and (3) all the unfinished tasks preceding T_j in the TPG satisfy the above condition (2).

3.5 Optimization variables and objectives

MODSPSP's optimization variables and objectives at a specific rescheduling point are formulated in this section. At the rescheduling point t_l ($t_l > t_0$), considering all the current information gathered from the software project, which contains attributes of a set of available employees $e\_ava\_set(t_l)$, a set of

available tasks $T\_ava\_set(t_l)$ with their remaining estimated task efforts, and the TPG $G(V(t_l), A(t_l))$ updated at t_l, MODSPSP consists in generating a new schedule $\mathrm{X}(t_l) = \left(x_{ij}(t_l)\right)_{M\times(N_1+N_{new}(t_l))}$ representing the dedication matrix of each employee to each task by optimizing the following objectives:

$$\min \mathbf{F}(t_l)=[f_1(t_l), f_2(t_l), f_3(t_l), f_4(t_l), f_5(t_l)] \tag{2}$$

where $x_{ij}(t_l)$ indicates the dedication of employee e_i to task T_j scheduled at time t_l. It measures the percentage of a full-time job which e_i spends on T_j. The objectives $f_1(t_l)$, $f_2(t_l)$, $f_3(t_l)$, $f_4(t_l)$ and $f_5(t_l)$ are related to the project duration, project cost, schedule robustness, stability of the project, and employees' satisfaction, respectively. The formulae of each objective are given below.

$$f_1(t_l) = duration_I = \max_{\{j|T_j\in T\_ava\_set(t_l)\}}\left(T_j^{end}(t_l)\right) - \min_{\{j|T_j\in T\_ava\_set(t_l)\}}\left(T_j^{start}(t_l)\right) \tag{3}$$

The duration measure $f_1(t_l)$ in (3) evaluates the maximum elapsed time required for completing the remaining effort of each available task rescheduled at t_l. The subscript I in $duration_I$ represents the initial scenario, which assumes no task effort variances. $T_j^{start}(t_l)$ denotes the time at which the remaining effort of T_j starts processing after t_l based on the new schedule, and $T_j^{end}(t_l)$ is the finishing time of T_j rescheduled at t_l.

$$f_2(t_l) = cost_I = \sum_{t'\geq t_l}\ \sum_{e_i\in e\_ava\_set(t_l)} e\_cost_i^{t'} \tag{4}$$

The cost measure $f_2(t_l)$ in (4) indicates the initial cost, which evaluates the total salaries paid to the available employees for their work on the available tasks at t_l, assuming no task effort uncertainties. Here, t' represents any month during which the project is being implemented after t_l, and $e\_cost_i^{t'}$ denotes the salaries paid to employee e_i at the moment of time t'. $e\_cost_i^{t'}$ is calculated as follows:

If $\sum_{j\in T\_active\_set(t')} x_{ij}(t_l) \leq 1$, then

$$e\_cost_i^{t'} = e_i^{norm\_salary}\cdot t'\cdot \sum_{j\in T\_active\_set(t')} x_{ij}(t_l) \tag{5}$$

else if $1< \sum_{j\in T\_active\_set(t')} x_{ij}(t_l) \leq e_i^{maxded}$, then

$$e\_cost_i^{t'} = e_i^{norm\_salary}\cdot t'\cdot 1 + e_i^{over\_salary}\cdot t'\cdot\left(\sum_{j\in T\_active\_set(t')} x_{ij}(t_l) - 1\right) \tag{6}$$

If e_i works overtime at t' (the total dedications of e_i to all the active tasks at t' are larger than 1), then the overtime salary $e_i^{over\_salary}$ should be paid for the overtime working. The salary $e_i^{norm\_salary}$ is paid for normal working time.

$$f_3(t_l) = robustness = \sqrt{\frac{1}{N}\sum_{q=1}^{N}\left(\max\left(0, \frac{duration_q(t_l) - duration_I(t_l)}{duration_I(t_l)}\right)\right)^2} + \lambda\sqrt{\frac{1}{N}\sum_{q=1}^{N}\left(\max\left(0, \frac{cost_q(t_l) - cost_I(t_l)}{cost_I(t_l)}\right)\right)^2} \quad (7)$$

The robustness measure $f_3(t_l)$ in (7) evaluates the sensitivity of a schedule's efficiency quality to task effort variances based on a scenario-based approach. Here, $duration_I$ and $cost_I$ are the initial duration and cost obtained from (3) and (4), $duration_q$ and $cost_q$ are the corresponding efficiency objective values under the q^{th} sampled task effort scenarios. N is the sample size, and λ is a weight parameter. In our experiments, we set N=30, and $\lambda = 1$.

$$f_4(t_l) = stability = \sum_{\{i|e_i \in e\_ava\_set(t_{l-1}) \cap e\_ava\_set(t_l)\}} \sum_{\{j|T_j \in T\_ava\_set(t_{l-1}) \cap T\_ava\_set(t_l)\}} \omega_{ij} \left| x_{ij}(t_l) - x_{ij}(t_{l-1}) \right| \quad (8)$$

The stability measure $f_4(t_l)$ in (8) calculates the weighted sum of dedication deviations between the new and original schedules. It assists in preventing employees from being shuffled around too much, and is evaluated for all the available tasks at t_l ($t_l > t_0$) which are left from the previous schedule generated at t_{l-1}. We set the weight ω_{ij} as shown in (9):

$$\omega_{ij} = \begin{cases} 2 & \text{if } x_{ij}(t_{l-1})=0 \text{ and } x_{ij}(t_l) > 0 \\ 1.5 & \text{if } x_{ij}(t_{l-1}) > 0 \text{ and } x_{ij}(t_l) = 0 \\ 1 & \text{otherwise} \end{cases} \quad (9)$$

$$f_5(t_l) = satisfaction = \frac{\sum_{e_i \in e\_ava\_set(t_l)} \sum_{T_j \in T\_ava\_set(t_l)} x_{ij}(t_l) \cdot \frac{\sum_{k \in req_j \cap skill_i} \left(1 - ED_i^k\right)}{\left| req_j \cap skill_i \right|}}{\left| e\_ava\_set(t_l) \right|} \quad (10)$$

The satisfaction measure $f_5(t_l)$ given in (10) evaluates the average degree of unwillingness of the employees to engage with the allocated tasks weighted by the dedication of employees to tasks. This objective is required to be minimized, as the other four objectives. The smaller the value of $f_5(t_l)$, the better the employees' satisfaction with the generated schedule.

The pseudo code for evaluating $f_1(t_l)$, $f_2(t_l)$, $f_3(t_l)$ and $f_4(t_l)$ is same as the one provided in our previous work [35], except that when calculating the objectives of project duration and cost, the

proficiency $e_{ij}^{Proficiency}(t_l)$ of employee e_i for task T_j is updated at each t_l according to e_i^{LA}, e_i^{MO} and SD^k.

It is worth mentioning that at the initial time t_0, only four of the objectives defined above ($duration_I$, $cost_I$, *robustness* and *satisfaction* (without *stability*)) are optimized.

3.6 Constraints

In MODSPSP, the search space constraints at the rescheduling point t_l are the following:

1) No overwork constraints

At the moment of time t' after t_l, the total dedication of an available employee to all the active tasks that are being developed should not exceed his/her maximum dedication to the project, i.e.,

$$\forall\, e_i \in e\_ava\_set(t_l),\ \forall t' \ge t_l,\quad e\_work_i^{t'} = \sum_{j \in T\_active\_set(t')} x_{ij}(t_l),\ \text{s.t.}\ \ e\_work_i^{t'} \le e_i^{maxded} \tag{11}$$

2) Task skill constraints

All the available employees working together for one available task must collectively cover all the skills required by that task, i.e.,

$$\forall\, T_j \in T\_ava\_set(t_l),\ \text{s.t.}\ \ req_j \subseteq \bigcup_{e_i \in e\_ava\_set(t_l)} \left\{ skill_i \mid x_{ij}(t_l) > 0 \right\} \tag{12}$$

3) Maximum headcount constraints

$$\forall\, T_j \in T\_ava\_set(t_l),\ \text{s.t.}\ \ T_j^{teamsize}(t_l) \le \max\left(T_j^{maxhead}, T_j^{\min\_num_{emp}}(t_l) \right) \tag{13}$$

where $T_j^{teamsize}(t_l)$ is the team size for accomplishing task T_j, $T_j^{\min\_num_{emp}}(t_l)$ is the minimum number of available employees who should join T_j in order to satisfy the task skill constraint, and $T_j^{maxhead}$ is the desired upper limit for $T_j^{teamsize}(t_l)$, aiming to reduce the communication overhead. Here, $T_j^{maxhead}$ is estimated as provided by the authors of [3]: $T_j^{maxhead} = \max\left\{1,\ round\left(2/3\left(T_j^{est\_tot\_eff}\right)^{0.672}\right)\right\}$. In (13), if $T_j^{teamsize}(t_l)$ cannot be reduced to $T_j^{maxhead}$ without violating the task skill constraints (i.e., $T_j^{maxhead} < T_j^{\min\_num_{emp}}(t_l)$), then $T_j^{teamsize}(t_l)$ can be relaxed up to $T_j^{\min\_num_{emp}}(t_l)$.

3.7 Discussion

The MODSPSP model presented in our previous work [35] is an advanced version to that of the available ones in the literature. It captures more dynamic features of a real-world SPSP than previous models. When compared with our previous model, the superiorities of the improved MODSPSP model constructed in this paper are summarized as follows:

(1) Consideration of employees' properties on subjective initiative. Human resource is one of the main

resources for any software project. To emphasize the effects of human factors on project success, more properties on employees' initiative such as motivation, learning ability, and the degree with which each employee is willing to engage with each skill (and thus with each task), are introduced in the improved model.

(2) Improvement of the skill level. To be more consistent with the reality, the proficiency of each employee on each skill is allowed to be improved over time. Upon investigating a real-world software project, in our current model, employees' learning ability, motivation, and the skill difficulty are considered as the three most influencing factors on the skill proficiency growth rate. A relationship between these factors has been formulated and analyzed. In contrast, the skill level is regarded as time-invariant during the whole project in [35].

(3) Definition of the satisfaction objective. Based on the degree with which each employee is willing to engage with each skill (and thus with each task), the employees' satisfaction with the generated schedule has been considered together with project duration, cost, robustness and stability, highlighting the fact that employees' subjective initiative is paid more attention in modern software projects. In [35], the degree with which each employee is willing to engage with skills and the satisfaction of employees were not taken into account.

4 A Q-learning-based proactive-rescheduling approach to solve the MODSPSP

4.1 The Q-learning-based proactive-rescheduling framework

4.1.1 Training agent to determine the appropriate scheduling approach using Q-learning

Q-learning is used to learn which scheduling approach is the most appropriate for the new project environment. The proposed Q-learning scheme for dynamic software project scheduling based on centralized control is shown in Fig.2. It is assumed that the agent is able to perceive information of the tasks and employees in the project environment, and make an appropriate decision about the action to be taken in the new environment. Here, the action refers to the use of certain global and local search methods in the MOTAMAQ-based scheduling approach. The software project is regarded as the environment for carrying out Q-learning. The MOTAMAQ-based scheduling approach follows the instructions of the agent. After a set of non-dominated scheduling solutions are generated by the specified scheduling approach, a signal is sent to the agent. Reward for the selected action is evaluated by the agent, and the corresponding state-action value is updated. The agent would give the instruction of the next action which is selected based on a predefined selection policy. Afterwards, for each state of the software project, the state-action value converges to an optimum value in iterative runs.

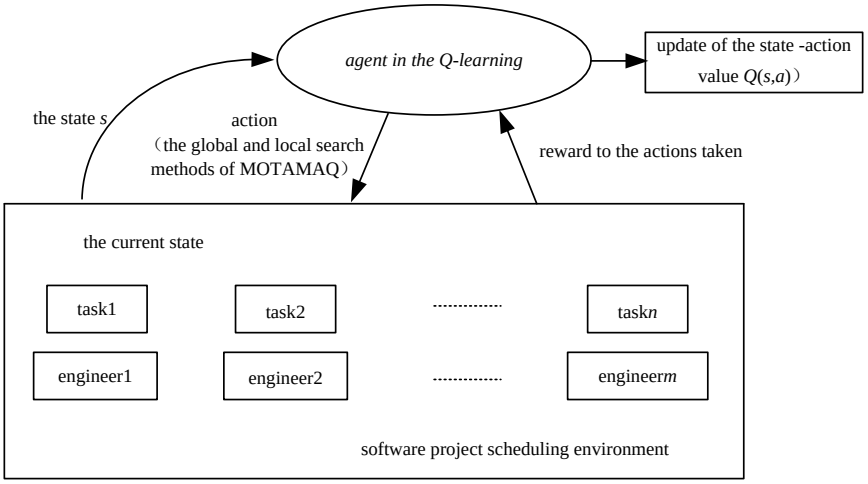

Fig. 2. Q-learning scheme for dynamic software project scheduling

4.1.2 Procedure of the Q-learning-based proactive-rescheduling framework

To handle both uncertainties and real-time events occurring during a software project, a Q-learning-based proactive-rescheduling framework is proposed for solving MODSPSP. Its pseudocode is depicted in Fig. 3.

Procedure 1 The Q-learning-based proactive-rescheduling framework
******** Initialization *********
1: Set $l = 0$.
2: All the Q-values in the state-action pair table are initialized to be 0.
3: The initial state $S(t_l)$ of the project environment is perceived by the agent.
4: Select an arbitrary action $A(t_l)$. The global and local search methods of MOTAMAQ are selected based on the action $A(t_l)$.
******** Proactive scheduling *********
5: MOTAMAQ is triggered, and automatically generates a set of non-dominated solutions to optimize the four objectives, i.e., duration, cost, robustness, and employees' satisfaction, satisfying the three constraints defined by (11) - (13).
6: Calculate the HV value of the obtained non-dominated solution set, and set it as the reward $r(t_l)$.
7: The software manager selects one solution manually, or based on an automated decision making procedure.
******** Rescheduling *********
8: while the project is not completed
9: The new generated schedule is implemented in the current project.
10: if a critical dynamic event occurs
11: $l = l + 1$.
12: The skill level of each employee is updated.
13: The current state $S(t_l)$ of the project environment as the result of executing action $A(t_{l-1})$ is perceived by the agent.
14: The value of $Q(S(t_{l-1}), A(t_{l-1}))$ is updated according to (23) in Section 4.8.
15: An action $A(t_l)$ is chosen based on the selection policy given in Section 4.6.
16: The MOTAMAQ-based rescheduling approach determined by $A(t_l)$ is triggered and automatically generates a set of non-dominated solutions, which represent different trade-offs among the five objectives: duration, cost, robustness, stability and satisfaction, satisfying the three constraints defined by (11) - (13).
17: The reward value $r(t_l)$ is calculated to evaluate the performance of $A(t_l)$.
18: The software manager decides one schedule from the generated non-dominated solution set.
19: end if
20: end while
21: Exit.

Fig. 3. Pseudo code for the Q-learning-based proactive-rescheduling framework.

4.2 MOTAMAQ-based rescheduling method for MODSPSP

Two_Arch2 [40] is a successful many-objective (4 or more objectives) optimization algorithm with low complexity. It maintains a convergence archive (CA) and a diversity archive (DA) to promote convergence

and diversity separately. It adopts the $I_{\varepsilon+}$ indicator [46] as the selection principle for CA to improve the convergence on many-objective optimization problems, and Pareto dominance for DA to promote diversity. In addition, it employs an *L*p-norm-based distance ($p < 1$) to maintain diversity in DA. MODSPSP is a dynamic problem with five objectives. To solve it in an efficient way, our scheduling approach MOTAMAQ uses the general framework of Two_Arch2. Moreover, MOTAMAQ is also a memetic algorithm that employs both global and local search to generate child individuals. The appropriate global and local search methods in a specific environment are learned adaptively by the agent in Q-learning.

At each rescheduling point t_l ($t_l > t_0$), a MOTAMAQ-based rescheduling approach is trigged to obtain a new schedule in the new environment. The pseudo code for MOTAMAQ is presented in Fig. 4.

Procedure 2 MOTAMAQ at the rescheduling point t_l ($t_l > t_0$)

Input: n_{POP} - the population size. n_{CA} - the fixed size of the convergence archive $CA(t_l)$. n_{DA} - the fixed size of the diversity archive $DA(t_l)$. Q - the number of uncertainty scenarios sampled. L_{max} - the maximum number of iterations of the local search. *NmbEvl* - the maximum number of objective vector evaluations. $A(t_l)$ - the action determined by Q-learning.

Output: $DA(t_l)$, *S*- a selected solution.

******** Initialization *********

1: Generate an initial population $P(t_l)$ using heuristic strategies according to the updated project state at t_l.

2: Sample a set of task effort scenarios θ_q at random according to the normal distribution, $q = 1,2,\ldots,Q$.

3: Evaluate each individual in $P(t_l)$.

4: All the Pareto non-dominated solutions are determined from $P(t_l)$ to form $DA(t_l)$.

5: Set $CA(t_l)$ as empty.

6: Set the counter of objective evaluation numbers $ct = |P(t_l)|$.

7: while $ct < NmbEvl$

******** Variation *********

8: Sample a set of task effort scenarios θ'_q at random according to the normal distribution, $q = 1,2,\ldots,Q$.

9: A certain global search method specified by $A(t_l)$ is employed on $CA(t_l)$ and $DA(t_l)$ to produce a child population $NPOP_1(t_l)$.

10: Evaluate each individual in $NPOP_1(t_l)$.

11: $ct = ct + |NPOP_1(t_l)|$.

12: A certain local search method specified by $A(t_l)$ is performed on the neighborhood of each individual in $NPOP_1(t_l)$ for L_{max} times, and a new child population $NPOP(t_l)$ is obtained.

13: $ct = ct + L_{\max} \cdot |NPOP_1(t_l)|$.

******** Update $CA(t_l)$ by the $I_{\varepsilon+}$ indicator *********

14: Find non-duplicated objective values of individuals in $CA(t_l) \cup NPOP(t_l)$, and set them as $CA(t_l)$.

15: if $|CA(t_l)| > n_{CA}$

16: The extra solutions are removed from $CA(t_l)$ according to the $I_{\varepsilon+}$-based fitness value.

17: end if

******** Update $DA(t_l)$ by Pareto dominance *********

18: Find non-dominated solutions in $NPOP(t_l) \cup DA(t_l)$, and set them as $DA(t_l)$.

19: if $|DA(t_l)| > n_{DA}$

20: Set $AD(t_l)$ empty.

21: while $|AD(t_l)| < n_{DA}$

22: Select an appropriate solution from $DA(t_l)$ based on the *Lp*-norm-based ($p < 1$) distance.

23: Add it to $AD(t_l)$.

24: end while

25: $DA(t_l) = AD(t_l)$.

26: end if

27: end while

28: Select one solution *S* from $DA(t_l)$ as the schedule to be implemented based on a decision making procedure.

29: Output $DA(t_l)$ and *S*.

30: Exit.

$|\cdot|$ means cardinality of a set

Fig. 4. Pseudo code for MOTAMAQ at the rescheduling point t_l ($t_l > t_0$).

In line 1 of the pseudo code shown in Fig. 4, updated project state is obtained in the same way as explained in section 4.2.2 of [35]. If genetic algorithm (GA)-based global search is performed, heuristic constructions of the initial population are obtained in the same way as explained in section 4.2.3 of [35]. If angle modulated differential evolution (AMDE) [14]-based global search is adopted, the population initialization is described in the following section 4.4.1 of this paper. In line 8, the sampled task efforts vary from one generation to another, which increases the probability of obtaining robust solutions undergoing a variety of scenarios. The global and local search methods in line 9 and line 12 are introduced in sections 4.4.1 and 4.4.2 of this paper, respectively. Updates of CA (lines 14-17) and DA (lines 18-26) are the same as those explained in [40]. The decision making procedure shown in line 28 is explained in section 4.2.4 of [35]. For each candidate solution, the constraint handling methods and the objective evaluation procedure are explained in sections 5.2 and 5.3 of [35], respectively.

It is worth noting that at the initial time t_0, the proactive scheduling is also performed using the MOTAMAQ procedure shown in Fig. 4, except that the population is randomly initialized in line 1 instead of using heuristic initialization, and when evaluating an individual, only four objectives (without stability) are considered.

4.3 State description

In MODSPSP, once a dynamic critical event occurs, a rescheduling approach is triggered. The new environment after the occurrence of a critical event is perceived by the agent, and is considered as the state in Q-learning. Since employees and tasks are the two main elements in the software project, the MODSPSP environment state is described by the following two aspects:

1) Effort ratio

It is defined as the ratio of the sum of remaining efforts of all the available tasks to the total efforts of all the tasks that have ever appeared in the project until now. This ratio is mainly affected by the number of available tasks and their remaining efforts.

$$fa_1 = Effort\ Ratio = \frac{\sum_{j \in T\_ava\_set(t_l)} T_j^{est\_rem\_eff}(t_l)}{\sum_{j=1}^{(N_I + N_{new}(t_l))} T_j^{est\_tot\_eff}} \quad (14)$$

2) Proficiency ratio

It is defined as the ratio of the sum of proficiencies of all the available employees for all the available tasks to the sum of the highest proficiencies (i.e., 1) of all the employees for all the available tasks. This ratio is mainly affected by the number of available employees and their respective proficiencies.

$$fa_2 = Proficiency\ Ratio = \frac{\sum_{j \in T\_ava\_set(t_l)} \sum_{i \in e\_ava\_set(t_l)} e_{ij}^{Proficiency}}{\sum_{j \in T\_ava\_set(t_l)} \sum_{i=1}^{M} 1} \quad (15)$$

Nine states are defined in our work. The criterion for the classification of the nine states is listed in Table 3.

Table 3

State classification.

S_1	S_2	S_3	S_4	S_5
$0 \le fa_1 \le 0.4$ $0 \le fa_2 \le 0.6$	$0 \le fa_1 \le 0.4$ $0.6 < fa_2 \le 0.9$	$0 \le fa_1 \le 0.4$ $0.9 < fa_2 \le 1$	$0.4 < fa_1 \le 0.8$ $0 \le fa_2 \le 0.6$	$0.4 < fa_1 \le 0.8$ $0.6 < fa_2 \le 0.9$
S_6	S_7	S_8	S_9	
$0.4 < fa_1 \le 0.8$ $0.9 < fa_2 \le 1$	$0.8 < fa_1 \le 1$ $0 \le fa_2 \le 0.6$	$0.8 < fa_1 \le 1$ $0.6 < fa_2 \le 0.9$	$0.8 < fa_1 \le 1$ $0.9 < fa_2 \le 1$	

4.4 Definition of actions

MOTAMAQ is a memetic algorithm, which is composed of both global and local searches. Different combinations global and local search methods are regarded as different actions.

4.4.1 Global search

Two global search methods: GA-based and AMDE-based are designed.

1) GA-based representations and variation operators

In MOTAMAQ's GA-based global search, binary encoding is used to represent individuals. The original solution of MODSPSP is a dedication matrix $\mathrm{X}(t_l) = \left(x_{ij}(t_l)\right)_{M \times (N_I + N_{new}(t_l))}$, where $x_{ij}(t_l) \in \left[0,\ e_i^{maxded}\right]$. Here, nb bits are employed to encode $x_{ij}(t_l)$; thus, $x_{ij}(t_l) \in \left\{0, e_i^{maxded} \cdot 1/k, \ldots, e_i^{maxded} \cdot k/k\right\}$, $k = 2^{nb} - 1$. The value of $x_{ij}(t_l)$ should be searched only when e_i and T_j are available at t_l; in any other cases, $x_{ij}(t_l) = 0$. For the sake of simpler computation, only the values of $x_{ij}(t_l) \in \left\{x_{ij}(t_l) \mid e_i^{available}(t_l) = 1 \text{ and } T_j^{available}(t_l) = 1\right\}$ are encoded, such that the chromosome has a length of $\left|e\_ava\_set(t_l)\right| \cdot \left|T\_ava\_set(t_l)\right| \cdot nb$ bits.

When evaluating the objective values, each chromosome should be decoded into a dedication matrix. An illustration of a binary chromosome representation and its decoded dedication matrix is shown in Fig. 5, where there are two available employees e_2 and e_3, two available tasks T_1 and T_2, one leaving employee e_1, one finished task T_3, and $nb = 3$.

In the GA-based global search, a crossover operator designed for matrices [27] is employed. It

decodes the parent binary chromosomes into dedication matrices at first. Then either rows or columns in the two parent dedication matrices are exchanged with an equal probability of 0.5. Later, the matrices are once again transformed into binary chromosomes. The mutation operator is bit-flip mutation.

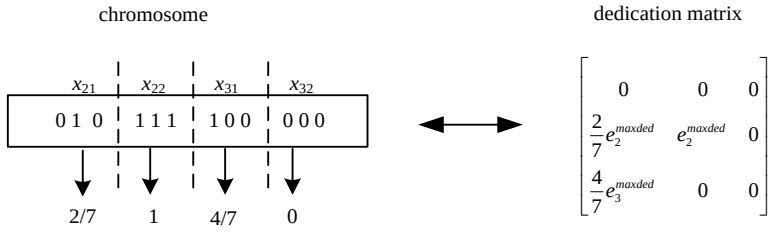

Fig. 5. An illustration of chromosome representation and its dedication matrix.

The pseudo code of GA-based global search is shown in Fig. 6. When the global search operator is set to GA-based global search, the procedure shown in Fig. 6 is called from line 9 of MOTAMAQ in Fig. 4 in order to produce an offspring population $NPOP_1(t_l)$.

Procedure 3 GA-based global search at the rescheduling point t_l ($t_l \geq t_0$)

Input: $CA(t_l)$ - the convergence archive. $DA(t_l)$ - the diversity archive. $P(t_l)$ - the initial population. n_{pop} - the population size.

Output: $NPOP_1(t_l)$

********Crossover*********

1: Set $NPOP_{11}(t_l)$ as empty.

2: While $|NPOP_{11}(t_l)| < 2 * n_{pop}$

3: If $CA(t_l)$ is empty

4: Select two individuals from $P(t_l)$ uniformly at random as the parents.

5: else

6: Select one parent individual from $DA(t_l)$ uniformly at random.

7: Select another parent individual from $CA(t_l)$. If $|CA(t_l)| = 1$, then select the individual in $CA(t_l)$ as the parent individual. Otherwise, two individuals are picked up uniformly at random from $CA(t_l)$, and check the domination of each other. If one dominates the other, the former will be chosen. Otherwise, one of them is selected at random.

8: end if

9: Perform the crossover operator designed for matrices on the two parent individuals. Then two child individuals are generated and added into $NPOP_{11}(t_l)$.

10: end while

********Mutation*********

11: Set $NPOP_{12}(t_l)$ empty.

12: while $|NPOP_{12}(t_l)| < n_{pop}$

13: if $CA(t_l)$ is empty

14: Select one individual from $DA(t_l)$ uniformly at random.

15: else

16: Select one individual from $CA(t_l)$ uniformly at random.

17: end if

18: Perform the bit-flip mutation on the selected individual, and a child individual is generated and added into $NPOP_{12}(t_l)$.

19: end while

20: $NPOP_1(t_l) = NPOP_{11}(t_l) \bigcup NPOP_{12}(t_l)$.

21: Output $NPOP_1(t_l)$.

22: Exit.

Fig. 6. Pseudo code for GA-based global search at the rescheduling point t_l ($t_l \geq t_0$).

2) **AMDE-based representations and variation operators**

AMDE is designed for solving binary-valued optimization problems through operations from the original DE. It converts a binary-valued problem into a 4-dimensional problem in a continuous-valued space using angular modulation. AMDE operates by evolving the values of the four coefficients *a*, *b*, *c*, and *d* of the bit string generating function given by (16).

$$g(x) = \sin(2\pi(x-a)\times b \times \cos(2\pi(x-a)\times c)) + d \quad (16)$$

Hence, in our AMDE-based global search, 4-dimensional real number encoding has been used for individual representation, with the range of $[-1, 1]$ for each dimension [14].

In (16), $x \in [-2, 2]$ has been selected as shown in [14]. Similar to the GA-based global search, the binary chromosome has a length of $l_1 = |e\_ava\_set(t_l)| \cdot |T\_ava\_set(t_l)| \cdot nb$ bits. Thus, l_1 values for *x* are evenly sampled from $[-2, 2]$. The decoding procedure is described as follows: when a new individual with 4 dimensions is evolved using differential evolution (DE) [26] operators, replace the coefficients *a*, *b*, *c*, and *d* from (16) by its values. The resulting function value is calculated for each of the l_1 sampled values iteratively. If the result is positive, a bit-value of 1 is recorded; otherwise, a bit-value of 0 is recorded. Thus a bit string with the length of l_1 is obtained, and then translated into a dedication matrix as shown in Fig. 5 and evaluated. The DE mutation operators used in this study are DE/rand/2 and DE/curr. to best/1 [26].

If AMDE-based global search is used, the initial population in line 1 of Fig. 4 is constructed as follows: to use the history information of MODSPSP and speed up algorithm convergence, 20% of the initial population are formed with the historic scheduling solution (the individual in the continuous-valued space with 4 dimensions) from the last rescheduling point and its variants produced by polynomial mutation [10]. The remaining 80% of the population is filled with random individuals.

The pseudo code of AMDE-based global search is shown in Fig. 7. When the global search operator is set to AMDE-based global search, the procedure shown in Fig. 7 is called from line 9 of MOTAMAQ in Fig. 4 in order to produce an offspring population $NPOP_1(t_l)$.

Procedure 4 AMDE-based global search at the rescheduling point t_l ($t_l \geq t_0$)

Input: $CA(t_l)$, $DA(t_l)$, $P(t_l)$, n_{pop}, F_1, F_2, F_3-mutation factors, *CR*-crossover ratio

Output: $NPOP_1(t_l)$

******** DE/rand/2 mutation and binomial-crossover*********

1: Set $NPOP_{11}(t_l)$ as empty.

2: while $|NPOP_{11}(t_l)| < 2 * n_{pop}$

3: if $CA(t_l)$ is empty

4: Select one individual from $P(t_l)$ uniformly at random as a target individual *p*. If $|DA(t_l)| > 1$, select p_{r_1}, p_{r_2} and p_{r_3} from

$P(t_l) \setminus p$, and p_{r_4} and p_{r_5} from $DA(t_l)$ uniformly at random, respectively; else if $|DA(t_l)| = 1$, select p_{r_1}, p_{r_2}, p_{r_3} and p_{r_4} from $P(t_l) \setminus p$, and p_{r_5} from $DA(t_l)$ uniformly at random, respectively.

5: else

6: Select one individual from $CA(t_l)$ uniformly at random as a target individual *p*. If $|CA(t_l)| > 3$ and $|DA(t_l)| > 1$, select p_{r_1}, p_{r_2} and p_{r_3} from $CA(t_l) \setminus p$, and p_{r_4} and p_{r_5} from $DA(t_l)$ uniformly at random, respectively; else select p_{r_1}, p_{r_2}, p_{r_3}, p_{r_4} and p_{r_5} from $(CA(t_l) \cup DA(t_l) \cup P(t_l)) \setminus p$ uniformly at random.

7: end if

8: The DE/rand/2 mutation is performed on *p*, and a donor solution *v* is generated: $v = p_{r_1} + F_1(p_{r_2} - p_{r_3} + p_{r_4} - p_{r_5})$.

9: Check each dimension of *v*. If the value of a certain dimension is out of bounds, it is replaced by a random value within bounds.

10: Apply the binomial crossover operator on *p* and *v*, to obtain a trial solution *u*: $u_j = \begin{cases} v_j & \text{if } rand_j \le CR \text{ or } j = j_{rand} \\ p_j & \text{else} \end{cases}$, where u_j, v_j, and p_j are the *j*th dimension of *u*, *v*, and *p*, respectively, $j = 1,2,3,4$, and j_{rand} is a value generated from $\{1,2,3,4\}$ uniformly at random.

11: *u* is added into $NPOP_{11}(t_l)$.

12: end while

******** DE/curr. to best/1 mutation and binomial-crossover*********

13: Set $NPOP_{12}(t_l)$ empty.

14: while $|NPOP_{12}(t_l)| < n_{pop}$

15: if $CA(t_l)$ is empty

16: Select one individual from $P(t_l)$ uniformly at random as a target individual *p*. Select p_{r_2} and p_{r_3} from $P(t_l) \setminus p$, and p_{best} from $DA(t_l)$ uniformly at random, respectively.

17: else

18: Select one individual from $CA(t_l)$ uniformly at random as a target individual *p*, and select p_{best} from $DA(t_l)$ uniformly at random. If $|CA(t_l)| > 2$, select p_{r_2} and p_{r_3} from $CA(t_l) \setminus p$ uniformly at random, else, select p_{r_2} and p_{r_3} from $(CA(t_l) \cup P(t_l)) \setminus p$ uniformly at random.

19: end if

20: The DE/ curr. to best/1 mutation is performed on *p*, and a donor solution *v* is generated: $v = p + F_2(p_{r_2} - p_{r_3}) + F_3(p_{best} - p)$.

21: Check each dimension of *v*. If the value of a certain dimension is out of bounds, it is replaced by a random value within bounds.

22: Apply the binomial-crossover operator on *p* and *v*, to obtain a trial solution *u*.

23: *u* is added to $NPOP_{12}(t_l)$.

24: end while

25: $NPOP_1(t_l) = NPOP_{11}(t_l) \cup NPOP_{12}(t_l)$.

26: Output $NPOP_1(t_l)$.

27: Exit.

Fig. 7. Pseudo code for DE-based global search at the rescheduling point t_l ($t_l \ge t_0$).

4.4.2 Local search

To work with the GA-based global search (for which the individual is defined in the binary-valued

space), two local search approaches are designed, which are denoted as LS_1^b and LS_2^b. With LS_1^b, the value of each entry in the dedication matrix $X(t_l)$ of the considered individual is replaced by a different value selected uniformly at random from the set $\{0, e_i^{maxded} \cdot 1/k, \cdots, e_i^{maxded} \cdot k/k\}$ with a probability P_m, where $P_m = 1/(|T\_ava\_set(t_l)| \cdot |e\_ava\_set(t_l)|)$. LS_2^b swaps two randomly selected rows or columns in $X(t_l)$. It is worth noting that after performing the above local search, the heuristic operator is executed on the dedication matrix of the resulting solution, which sets the dedication of an employee for a task to 0 if he/she has none of the skills required by the task, i.e., if $e_{ij}^{Proficiency} = 0$, then set $x_{ij}(t_l) = 0$.

To work with the AMDE-based global search (where the individual is defined in the continuous-valued space), two local search approaches are designed that are denoted by LS_1^c and LS_2^c. In LS_1^c, polynomial mutation is performed on each entry of the considered individual with a probability $P_m^{'} = 1/4$ (the length of an individual in AMDE is 4), whereas for LS_2^c, uniform mutation is performed with a probability of $P_m^{'}$. The heuristic operator is also performed on the dedication matrix of the resulting solution.

4.4.3 Four actions

A total of four actions are defined by combining the global search procedures described in section 4.4.1 with the local search procedures described in section 4.4.2. These actions are listed in Table 4.

Table 4

The four MOTAMAQ actions.

A_1	A_2	A_3	A_4
GA+LS_1^b	GA+LS_2^b	AMDE+LS_1^c	AMDE+LS_2^c

4.5 State-action pair table

The state-action pair table is presented in Table 5. This table is used by Q-learning to decide which action to use for a given state as explained in section 4.6. Its *Q*-values are updated as explained in sections 4.7 and 4.8.

Table 5

State-action pair table.

Action No. / State No.	A_1	A_2	A_3	A_4
S_1	$Q(S_1, A_1)$	$Q(S_1, A_2)$	$Q(S_1, A_3)$	$Q(S_1, A_4)$
S_2	$Q(S_2, A_1)$	$Q(S_2, A_2)$	$Q(S_2, A_3)$	$Q(S_2, A_4)$
S_3	$Q(S_3, A_1)$	$Q(S_3, A_2)$	$Q(S_3, A_3)$	$Q(S_3, A_4)$
……	……	……	……	……
S_9	$Q(S_9, A_1)$	$Q(S_9, A_2)$	$Q(S_9, A_3)$	$Q(S_9, A_4)$

4.6 Action selection policy

For the perceived environment state $S(t_l)$, a selection policy $\pi(S(t_l))$, is used by an agent to select an action. Here, the selection probability $P(S(t_l), A_i)$ for each candidate action A_i ($i = 1, 2, \cdots, NA$, where *NA* is the number of candidate actions), is determined by the *Q*-values in the state-action pair table with the help of the *softmax* function:

$$P(S_{t_l}, A_i) = \frac{exp\left(\theta Q(S(t_l), A_i) / \max\limits_{i=1,2,\cdots,NA} Q(S(t_l), A_i)\right)}{\sum_{i=1}^{NA} exp\left(\theta Q(S(t_l), A_i) / \max\limits_{i=1,2,\cdots,NA} Q(S(t_l), A_i)\right)} \tag{17}$$

where, $\theta = 2$ is used in this work. An action $A(t_l)$ is then chosen from $\{A_1, A_2, \cdots, A_{NA}\}$ according to the roulette wheel selection.

4.7 Reward of an action

The reward value is essentially served to reinforce the action and guide the agent to accomplish its goal. After executing $A(t_l)$, a reward value $r(t_l)$ is given to evaluate the performance of $A(t_l)$ and also to update the state-action pair value $Q(S(t_l), A(t_l))$. For multi-objective optimization, the *hypervolume* (*HV*) [47] is a commonly used metric mainly because of its ability to evaluate both the convergence and spread of the obtained non-dominated front. In our approach, *HV* is calculated as the reward $r(t_l)$ for the selected action $A(t_l)$. A larger $r(t_l)$ value indicates a better convergence and a wider spread.

The reference point for calculating *HV* is obtained by estimating the worst value on each objective in the current environment state $S(t_l)$, which is given as follows:

$$\begin{aligned} f_1^{\max}(t_l) = duration_I^{\max} &= \sum_{T_j \in T\_ava\_set(t_l)} T_j^{est\_rem\_eff}(t_l) \Big/ \left(\min_{e_i \in e\_ava\_set(t_l)} e_i^{maxded} \Big/ k \Big/ \max_{T_j \in T\_ava\_set(t_l)} V_j \right) \\ &= 7k \cdot \sum_{T_j \in T\_ava\_set(t_l)} T_j^{est\_rem\_eff}(t_l) \Big/ \min_{e_i \in e\_ava\_set(t_l)} e_i^{maxded} \end{aligned} \tag{18}$$

In the worst case, tasks are processed one by one. The total dedication for each task is the minimum value $\min\limits_{e_i \in e\_ava\_set(t_l)} e_i^{maxded} / k$, and the cost driver value V_j of each task T_j takes the maximum value 7.

$$f_2^{\max}(t_l) = cost_I^{\max} = \sum_{e_i \in e\_ava\_set(t_l)} \sum_{T_j \in T\_ava\_set(t_l)} e_i^{over\_salary} \cdot T_j^{est\_rem\_eff}(t_l) \cdot 7 \tag{19}$$

In the worst case, all available employees are dedicated to all the tasks with individual overwork salary $e_i^{over\_salary}$. Besides, the total dedication of each employee to each task T_j is equal to the total effort

required for the task $T_j^{est\_rem\_eff}(t_l)\cdot 7$, in which the maximum cost driver value of each task takes the value 7.

$$f_3^{\max}(t_l) = robustness^{\max} = C_{rob} = 10 \tag{20}$$

From experimental observations, it is noted that the $robustness$ value of each scheduling solution is always a lot smaller than the constant C_{rob}.

$$f_4^{\max}(t_l) = stability^{\max} = \left|e\_ava\_set(t_l)\right| \cdot \left|T\_ava\_set(t_l)\right| \cdot 2 \cdot \max_{e_i \in e\_ava\_set(t_l)} e_i^{maxded} \tag{21}$$

In the worst case, the dedication deviation of each available employee to each available task is $\max_{e_i \in e\_ava\_set(t_l)} e_i^{maxded}$, and the weight ω_{ij} always takes the maximum value of 2.

$$f_5^{\max}(t_l) = SD^{\max} = \left|T\_ava\_set(t_l)\right| \cdot \max_{e_i \in e\_ava\_set(t_l)} e_i^{maxded} \tag{22}$$

In the worst case, the average dissatisfaction degree and the dedication of each available employee to each allocated available task are 1 and $\max_{e_i \in e\_ava\_set(t_l)} e_i^{maxded}$, respectively.

4.8 Update of the Q-value

During the learning process of the agent in *Q*-learning, the *Q*-value of the state-action pair $Q(S(t_{l-1}), A(t_{l-1}))$ is updated as follows:

$$Q(S(t_{l-1}), A(t_{l-1})) = (1-\alpha)Q(S(t_{l-1}), A(t_{l-1})) + \alpha\left[r(t_{l-1}) + \gamma \max_{A_i \in \{A_1, A_2, \cdots, A_{NA}\}} Q(S(t_l), A_i)\right], \quad t \geq 1 \tag{23}$$

where $\max_{A_i \in \{A_1, A_2, \cdots, A_{NA}\}} Q(S(t_l), A_i)$ is the maximum state-action pair *Q*-value at the new perceived state $S(t_l)$ after executing $A(t_{l-1})$; $0 \leq \alpha \leq 1$ is the learning rate; $0 \leq \gamma \leq 1$ is the discount rate, which indicates the influence of the future reward on the current situation.

5 Experimental studies

Considering the complex and dynamically changing environments of software projects, we perform experiments with the aim of providing software managers with a detailed insight on selecting a scheduling approach for solving MODSPSP. This insight should be supported by evidences demonstrating which scheduling approach is likely to behave better based on the performance indexes that may affect the software manager's decision. Therefore, in this section, we compare our Q-learning based scheduling approach (MOTAMAQ) with seven other MOEA-based dynamic scheduling approaches in terms of convergence, distribution, and spread performance metrics. These performance metrics are often used to evaluate multi-objective optimization approaches. In addition, different trade-offs among the five objectives are analyzed by presenting the Pareto fronts of software projects.

5.1 Dynamic software project simulation model and instances

In order to validate the effectiveness and efficiency of our proposed model and approach, 18

MODSPSP instances derived from Alba and Chicano's benchmarks [1], and 3 real-world instances derived from business software construction projects for a departmental store [4] are adopted in this work. All the experiments are implemented in MATLAB running on a personal computer with Intel core i5, 3.2 GHz CPU and 4 GB RAM.

The 21 MODSPSP test instances generated here are similar to those used in our previous work [35]. The differences between them are summarized as follows: (1) Properties such as learning ability e_i^{LA}, motivation e_i^{MO}, and the degree with which employees are willing to engage with each skill e_i^{ED} are attached to each employee e_i ($i = 1, 2, \ldots, M$), and the degree of difficulty SD^k is attributed to the k^{th} skill ($k = 1, 2, \ldots, S$). (2) The skill level of each employee can be improved with time according to Eq. (1) such that the proficiency of each employee for each task is also improved. The range of the proficiency score is [0, 100]. From interviews with real-world software managers, for most people, the learning ability and motivation factors are close to the average level, and the probability of possessing a higher or lower level decreases gradually. Thus, e_i^{LA} and e_i^{MO} are assumed to follow the normal distribution N(0.5, 0.15), with a mean value of 0.5 and variance of 0.15. SD^k and e_i^{ED} are sampled uniformly from [0, 1] at random, respectively. If an employee possesses one skill, the initial proficiency score is sampled uniformly from (0, 100] at random, otherwise it is set to 0. For all 21 test instances, 10 new tasks are added one by one following the Poisson distribution during the project implementation. Besides, employee leaves and returns are also assumed to follow the Poisson distribution.

The 18 randomly generated MODSPSP instances are named as sT#1_dT#2_E#3_SK#4-#5, where sT#1 represents the number of initial static tasks, dT#2 denotes the number of new arriving tasks, E#3 represents the number of employees, and SK#4-#5 denotes each employee possesses #4 to #5 skills. For example, sT20_dT10_E10_SK4-5 means that 20 tasks exist initially in the project; then 10 new tasks are added one by one dynamically. A total of 10 employees each of whom possesses 4-5 skills are available to take part in the project. The 3 real-world instances are named Real_1, Real_2 and Real_3, respectively.

5.2 Parameter settings

Parameter settings of our approach MOTAMAQ are presented in Table 6. For each independent run, the algorithm iterates until 15000 objective vector evaluations are performed. In order to decide which solution to adopt for a given rescheduling point, the decision-making procedure presented in [35] is adopted with the following pairwise comparison matrix for the five objectives:

$$\mathrm{C}_1 = \left(c_{ij}\right)_{5\times 5} = \begin{bmatrix} 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 \\ 1/2 & 1/2 & 1 & 1 & 1 \\ 1/2 & 1/2 & 1 & 1 & 1 \\ 1/2 & 1/2 & 1 & 1 & 1 \end{bmatrix}.$$

Thus, the corresponding weight vector for the objectives is:

$$\mathrm{w} = \left(w_i\right)_{5\times 1} = [0.2857\ 0.2857\ 0.1429\ 0.1429\ \ 0.1429]^T .$$

Table 6

Parameter settings of MOTAMAQ.

n_{POP} - population size	100
n_{CA} - the fixed size of $CA(t_l)$	100
n_{DA} - the fixed size of $DA(t_l)$	100
Q - the number of uncertainty scenarios sampled	30
L_{max} - the iteration number of local search	5
p - the parameter in the Lp-norm-based distance	1/4 in the proactive scheduling, 1/5 in the rescheduling procedure
Crossover probability in GA-based global search	0.9
Mutation probability in GA-based global search	$1/L$, where L is the chromosome length
CR - crossover ratio in AMDE-based global search	0.1
F_1, F_2, F_3 - mutation factors in AMDE-based global search	a random number sampled uniformly from [0.8, 0.9]
maximum number of objective vector evaluations	15000

5.3 Validating the effectiveness of the strategies designed in MOTAMAQ

5.3.1 Introduction to the compared approaches

In this section, the proposed Q-learning-based scheduling approach MOTAMAQ is compared with seven other MOEA-based dynamic scheduling approaches. These other approaches also used the framework of proactive-rescheduling, i.e., a robust schedule is generated initially by proactive scheduling with regards to project uncertainties, and then the previous schedule is revised by the rescheduling approach in response to critical dynamic events. The heuristic population initialization mechanism that has been validated as an effective dynamic optimization strategy [35] is also adopted in the rescheduling approach. The differences among the compared algorithms are in that the proactive scheduling and rescheduling approaches are based on different MOEAs.

To validate the effectiveness of the many-objective handling strategy, the local search operators, and the Q-learning based learning mechanism employed in MOTAMAQ, the approach is compared to dε-MOEA (proposed as a rescheduling method in our previous work [35]), a MOEA/D-DE [23]-based rescheduling method, and a NSGA-III [8]-based rescheduling method. The dε-MOEA uses the ε-domination relation [9] and adopts efficient parent and archive update strategies. It has been validated as effective in producing good convergence and diversity compared to the state-of-the-art dynamic MOEA [35]. MOEA/D [44] is a promising algorithm which has gathered significant attention in recent years. It provides a new and general framework for solving multi-objective or many-objective problems based on decomposition. MOEA/D-DE is an improved version of MOEA/D, which employs a DE operator for generating new individuals. NSGA-III is a recently developed reference-point-based many objective evolutionary algorithm following the NSGA-II [11] framework. It maintains diversity based on a set of uniformly distributed reference points assigned in advance. The chromosome representations and variation operators in dε-MOEA and NSGA-III are the same as those in the GA-based global search, and the same parameter settings of

MOTAMAQ are used with dε-MOEA. Parameter settings of NSGA-III are: in the proactive scheduling (four objectives are considered), the number of reference points is 165, and the population size is 168. In the rescheduling approach (five objectives are considered), the number of reference points is 210, and the population size is 212. The chromosome representations in MOEA/D-DE are the same as that in AMDE-based global search, and the DE operator is represented in the similar fashion as in [23]. Parameter settings of MOEA/D-DE are: T (the neighborhood size of each subproblem) is 5, δ (the probability that the parent individuals in variation operators are selected from the neighborhood) is 0.9 and η_r (the maximal number of individuals replaced in the neighborhood when updated) is 2 [23].

To further investigate the influence of the self-adaptive learning mechanism based on Q-learning, MOTAMAQ has been compared to four additional algorithms without Q-learning. These algorithms use each of the four actions listed in section 4.4.3, respectively. In other words, the global and local search methods of these algorithms are fixed to a single action while executing the entire project, no matter how the project environment changes. The four algorithms are named as MOTAMA-GA- LS_1^b, MOTAMA-GA- LS_2^b, MOTAMA-AMDE- LS_1^c, MOTAMA-AMDE- LS_2^c, respectively. This group of comparisons can provide a software manager with an insight into whether it would be helpful to learn the features of different environments, and select the appropriate search operators adaptively according to the learned information. Parameter settings of the four algorithms are the same as those of MOTAMAQ (given in Table 6). Note that all algorithms will stop after 15000 objective vector evaluations in one run.

5.3.2 Performance measures

In multi-objective optimization, convergence, distribution and spread are the three main performance criteria used to evaluate the quality of the obtained Pareto front. If a Pareto front with good convergence, a uniform (in most cases) distribution and a wide spread can be found, the software manager can get a full picture of the various trade-offs among project duration, cost, robustness, stability, and satisfaction. Thus, he/she is able to make an informed decision or modify his/her own manual schedule on the basis of project requirements.

In this work, four popular metrics are used to evaluate the algorithm performances. The first one is the *hypervolume ratio* (*HVR*) [38] which calculates the ratio of the size of the objective space dominated by the obtained Pareto front PF_{known} to that dominated by the reference Pareto front PF_{ref}. A larger *HVR* value indicates a better convergence and a wider spread of the generated Pareto front. The second one is the *inverted generational distance* (*IGD*), which evaluates how far PF_{ref} is from PF_{known} [23]. *IGD* can measure both convergence and diversity. A small *IGD* value means the obtained solutions are close to PF_{ref} and do not miss any part of the whole PF_{ref}. The third one is a distribution metric named *Spacing*, which evaluates the distance variance of neighbouring vectors in PF_{known} [33]. A smaller *Spacing* indicates a more uniform distribution of PF_{known}. The fourth one is the modified *Spread* [35], which evaluates the extent of spread that the obtained solutions achieve and how uniform PF_{known} distributes in problems with more than two objectives. A small *Spread* indicates a wide and uniform spread of solutions in PF_{known}.

At each rescheduling point, the true Pareto front in the current environment is unknown in MODSPSP. Thus, PF_{ref} is obtained by merging the solutions produced in all the independent runs from a total of eight

algorithms, and then by determining the non-dominated solutions out of them. The reference point needed in *HVR* consists of the worst objective values obtained during all optimization runs. When deciding which algorithm to adopt, the convergence performance (*HVR* and *IGD*) of an algorithm should be considered first by the software manager, because better objective values are always vital. Out of two algorithms with equal amount of convergence, the one with a better distribution (*Spacing*) and spread (*Spread*) should be selected.

5.3.3 **Performance comparison procedure**

As previously mentioned, MOTAMAQ is compared with the seven other MOEA-based dynamic scheduling approaches in terms of the overall performance during the dynamic process of a project. For each MODSPSP instance, the procedure followed is described below:

Step 1: At the beginning of the project, MOTAMAQ is used as the proactive scheduling approach to generate a set of non-dominated schedules. Then a schedule is chosen to be implemented based on the decision-making procedure explained in [35].

Step 2: Once a critical dynamic event occurs, a rescheduling method is triggered. At each rescheduling point, the following sub-steps are carried out:

Sub-step 2.1: 30 independent runs of each approach are replicated. The "*robustness*" value is recalculated for each solution obtained by the eight approaches using the same 100 randomly sampled task efforts. Following this, eight updated non-dominated sets are obtained for each run.

Sub-step 2.2: All the updated non-dominated sets produced by the eight approaches in the 30 runs are merged. The new non-dominated solutions are determined from them to form the reference Pareto front.

Sub-step 2.3: For each approach in each of the 30 runs, the performance values (*HVR*, *IGD*, *Spacing*, *Spread*) are evaluated based on the reference Pareto front and its updated non-dominated set. Therefore, 30 values of each metric are recorded for each approach. As shown in Fig. 8, at the rescheduling point t_l, the 30 values are: $metric_j^{k,i}(t_l)$, $j=1,2,\ldots,30$, where $metric_j^{k,i}(t_l)$ represents the i^{th} performance metric value of the k^{th} approach in the j^{th} run at t_l, k=1,2,3,4,5,6,7,8, $i=1,2,3,4$, and *HVR*, *IGD*, *Spacing* and *Spread* are considered as the 1st to the 4th metric, and MOTAMAQ, dε-MOEA, MOEA/D-DE, NSGA-III, MOTAMA-GA- LS_1^b, MOTAMA-GA- LS_2^b, MOTAMA-AMDE- LS_1^c, MOTAMA-AMDE- LS_2^c are considered as the 1st to the 8th approach, respectively.

Sub-step 2.4: One solution is chosen from the reference Pareto front as the new schedule to be carried out in the project through the decision-making procedure explained in [35]. In this way, it will be ensured that, at each rescheduling point, the eight approaches are compared in the same project environment.

Step 3: If the entire project is not finished, then move to the next rescheduling point and go to Step 2; otherwise, go to Step 4.

Step 4: In order to compare the significance of the differences in the overall performances of the eight approaches across different runs and rescheduling points, Wilcoxon rank-sum tests with significance level of 0.05 are used in this work. For the j^{th} ($j=1,2,\ldots,30$) run of the k^{th} (k=1,2,3,4,5,6,7,8) approach, the i^{th}

($i=1,2,3,4$) performance values are averaged over the second half of the rescheduling points, as $mean_j^{k,i}$ shown in Fig. 8 (there is a training process for the learning of our approach, and to eliminate transient effects, only the latter half of the rescheduling points are considered to calculate the statistical performance values in dynamic environments). The 30 mean values $mean_j^{k,i}$ ($j=1,2,\ldots,30$) form the vector $Vec^{k,i}$. The $mean_j^{k,i}$ values are averaged first (the results are summarized in Table 7). Then, for the i^{th} metric, the pairwise comparisons between the vector $Vec^{1,i}$ of our approach and that of the other approach ($Vec^{k,i}$, k=2,3,4,5,6,7,8) are performed by Wilcoxon rank-sum tests. The results of the statistical tests are given in Table 8, and summarized in Table 9.

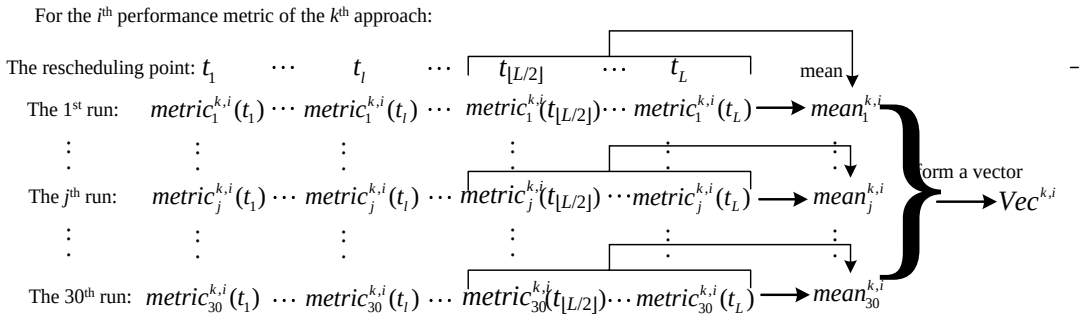

Fig. 8. An illustration for the overall performance comparisons of eight dynamic scheduling approaches in one MODSPSP instance (L is the total number of rescheduling points in the considered instance, and different instances may have different values of L).

Table 7

Average performance values of eight approaches across rescheduling points and different runs on the 21 test instances (The best value is in bold).

Metrics	*HVR*	*IGD*	*Spacing*	*Spread*	*HVR*	*IGD*	*Spacing*	*Spread*
Instance	sT10_dT10_E5_SK4-5				sT10_dT10_E10_SK4-5			
MOTAMAQ	**0.8351**	**0.1306**	0.0641	0.6216	**0.8235**	**0.1522**	0.0349	0.4838
dε-MOEA	0.7668	0.1965	0.0782	**0.5884**	0.7148	0.1736	**0.0340**	0.4926
MOEA/D-DE	0.7667	0.1987	0.0813	1.0672	0.3481	0.3889	0.0453	1.0112
NSGA-III	0.7931	0.1600	**0.0592**	0.6473	0.7020	0.1611	0.0352	0.4812
MOTAMA-GA-LS_1^b	0.7397	0.1614	0.0691	0.6381	0.7561	0.1560	0.0396	**0.4795**
MOTAMA-GA-LS_2^b	0.7283	0.1719	0.0770	0.6468	0.7326	0.1540	0.0378	0.5018
MOTAMA-AMDE-LS_1^c	0.7106	0.1747	0.0787	0.6467	0.5724	0.3451	0.0370	0.5367
MOTAMA-AMDE-LS_2^c	0.6983	0.1905	0.0725	0.6553	0.5651	0.3538	0.0350	0.5332
Instance	sT10_dT10_E15_SK4-5				sT10_dT10_E5_SK6-7			
MOTAMAQ	**0.8535**	**0.1425**	**0.0263**	0.5042	0.8433	**0.1193**	**0.0332**	0.4414
dε-MOEA	0.8087	0.1797	0.0351	0.5142	**0.8460**	0.1397	0.0389	0.4561
MOEA/D-DE	0.3625	0.3634	0.0425	1.0411	0.4025	0.3212	0.0513	0.9792
NSGA-III	0.7866	0.1770	0.0297	**0.4976**	0.7881	0.1567	0.0364	0.4190
MOTAMA-GA-LS_1^b	0.7200	0.1834	0.0326	0.5084	0.7673	0.1618	0.0377	0.4225
MOTAMA-GA-LS_2^b	0.7298	0.1914	0.0391	0.5482	0.7910	0.1531	0.0403	0.4410
MOTAMA-AMDE-LS_1^c	0.5728	0.3229	0.0332	0.6121	0.5600	0.2788	0.0433	**0.4020**
MOTAMA-AMDE-LS_2^c	0.5645	0.3343	0.0360	0.6204	0.5597	0.2889	0.0421	0.4113

Instance	sT10_dT10_E10_SK6-7				sT10_dT10_E15_SK6-7			
MOTAMAQ	**0.8388**	0.1359	**0.0351**	**0.4545**	**0.8817**	**0.1564**	0.0355	0.5054
dε-MOEA	0.8142	0.1890	0.0379	0.4633	0.8486	0.1925	0.0300	0.5243
MOEA/D-DE	0.5522	0.3505	0.0567	0.9667	0.2951	0.3920	0.0473	0.9285
NSGA-III	0.8209	**0.1299**	0.0372	0.4913	0.8109	0.2016	**0.0297**	**0.4803**
MOTAMA-GA-LS_1^b	0.7649	0.2044	0.0382	0.4569	0.8351	0.1644	0.0322	0.4935
MOTAMA-GA-LS_2^b	0.7869	0.1781	0.0364	0.4601	0.8442	0.1672	0.0369	0.4865
MOTAMA-AMDE-LS_1^c	0.6931	0.3150	0.0458	0.4818	0.6745	0.2477	0.0357	0.5339
MOTAMA-AMDE-LS_2^c	0.6882	0.3321	0.0463	0.4884	0.6714	0.2534	0.0342	0.5301
Instance	sT20_dT10_E5_SK4-5				sT20_dT10_E10_SK4-5			
MOTAMAQ	0.8595	**0.1372**	0.0373	0.4475	**0.8852**	**0.1982**	0.0325	0.5252
dε-MOEA	**0.8671**	0.1509	**0.0361**	**0.4206**	0.8525	0.2498	**0.0297**	0.5153
MOEA/D-DE	0.4582	0.3046	0.0529	0.9450	0.2486	0.4730	0.0497	0.9631
NSGA-III	0.8420	0.1608	0.0373	0.4288	0.8104	0.2437	0.0346	0.5292
MOTAMA-GA-LS_1^b	0.8208	0.1507	0.0386	0.4208	0.8201	0.2315	0.0377	0.5526
MOTAMA-GA-LS_2^b	0.8373	0.1478	0.0368	0.4359	0.8154	0.2320	0.0385	0.5190
MOTAMA-AMDE-LS_1^c	0.6804	0.2877	0.0396	0.4765	0.6320	0.3249	0.0357	0.4927
MOTAMA-AMDE-LS_2^c	0.6644	0.2942	0.0378	0.4830	0.6239	0.3153	0.0376	**0.4821**
Instance	sT20_dT10_E15_SK4-5				sT20_dT10_E5_SK6-7			
MOTAMAQ	**0.8860**	**0.2428**	0.0323	0.5610	**0.8792**	**0.1473**	0.0362	0.4441
dε-MOEA	0.8520	0.2783	0.0339	0.5788	0.8417	0.1648	0.0340	0.4397
MOEA/D-DE	0.2658	0.5692	0.0347	0.9743	0.3023	0.3905	0.0443	0.9514
NSGA-III	0.8027	0.2913	0.0314	0.5620	0.8206	0.1740	0.0349	0.4119
MOTAMA-GA-LS_1^b	0.8119	0.2861	**0.0262**	**0.5547**	0.7936	0.1536	0.0368	0.4130
MOTAMA-GA-LS_2^b	0.8142	0.2737	0.0280	0.5619	0.8008	0.1510	0.0377	**0.4085**
MOTAMA-AMDE-LS_1^c	0.5103	0.4190	0.0364	0.6328	0.6591	0.2989	**0.0338**	0.5314
MOTAMA-AMDE-LS_2^c	0.5131	0.4326	0.0360	0.6371	0.6507	0.3095	0.0359	0.5267
Instance	sT20_dT10_E10_SK6-7				sT20_dT10_E15_SK6-7			
MOTAMAQ	**0.8639**	**0.1832**	0.0490	0.5593	**0.8880**	**0.1879**	0.0283	0.5309
dε-MOEA	0.8272	0.2808	**0.0277**	0.4936	0.8573	0.2390	**0.0274**	0.5196
MOEA/D-DE	0.2899	0.3710	0.0426	0.8918	0.2901	0.5454	0.0405	0.9158
NSGA-III	0.8310	0.2371	0.0308	0.4943	0.8237	0.2491	0.0280	0.5122
MOTAMA-GA-LS_1^b	0.7795	0.2182	0.0326	**0.4645**	0.8293	0.2340	0.0300	0.5117
MOTAMA-GA-LS_2^b	0.7784	0.2223	0.0420	0.4778	0.8066	0.2283	0.0331	**0.5061**
MOTAMA-AMDE-LS_1^c	0.5718	0.2878	0.0454	0.5687	0.4905	0.3408	0.0369	0.6211
MOTAMA-AMDE-LS_2^c	0.5691	0.2858	0.0509	0.5802	0.4914	0.3431	0.0360	0.6263
Instance	sT30_dT10_E5_SK4-5				sT30_dT10_E10_SK4-5			
MOTAMAQ	0.7241	0.2040	**0.0274**	**0.5286**	0.8620	**0.1877**	0.0319	0.5166
dε-MOEA	**0.7318**	0.2567	0.0343	0.5777	**0.8888**	0.2482	0.0329	**0.4911**
MOEA/D-DE	0.3235	0.4799	0.0302	1.0215	0.5748	0.5532	0.0383	0.9574
NSGA-III	0.7309	**0.1973**	0.0288	0.5397	0.8671	0.2080	**0.0309**	0.5078
MOTAMA-GA-LS_1^b	0.7077	0.2442	0.0293	0.5378	0.8371	0.2322	0.0315	0.4918
MOTAMA-GA-LS_2^b	0.7246	0.2281	0.0319	0.5453	0.7973	0.2352	0.0352	0.5172
MOTAMA-AMDE-LS_1^c	0.4549	0.4068	0.0280	0.5553	0.5955	0.3320	0.0328	0.6213
MOTAMA-AMDE-LS_2^c	0.4409	0.4217	0.0282	0.5548	0.5964	0.3400	0.0333	0.6147
Instance	sT30_dT10_E15_SK4-5				sT30_dT10_E5_SK6-7			
MOTAMAQ	0.8072	0.2505	0.0311	0.5433	0.8888	**0.1617**	0.0380	0.5186
dε-MOEA	0.8151	0.2680	0.0306	0.5467	**0.8903**	0.2066	0.0356	0.4452

MOEA/D-DE	0.4361	0.4917	0.0440	0.9368	0.1768	0.5711	0.0375	0.9775
NSGA-III	0.7730	0.2801	0.0342	0.5582	0.8126	0.2201	0.0323	0.4617
MOTAMA-GA-LS_1^b	**0.8170**	**0.2479**	**0.0287**	**0.5275**	0.8004	0.2183	0.0378	**0.4438**
MOTAMA-GA-LS_2^b	0.7816	0.2498	0.0300	0.5468	0.8055	0.2284	0.0393	0.4592
MOTAMA-AMDE-LS_1^c	0.5901	0.5280	0.0393	0.6140	0.5830	0.4151	0.0381	0.4793
MOTAMA-AMDE-LS_2^c	0.5850	0.5251	0.0397	0.6213	0.5719	0.4208	**0.0309**	0.5953
Instance	sT30_dT10_E10_SK6-7				sT30_dT10_E15_SK6-7			
MOTAMAQ	0.8191	**0.2100**	0.0368	0.5372	**0.8323**	**0.2026**	**0.0255**	**0.5121**
dε-MOEA	**0.8533**	0.2558	**0.0271**	0.5008	0.8069	0.2203	0.0291	0.5684
MOEA/D-DE	0.2176	0.4838	0.0357	0.9495	0.2526	0.4755	0.0313	0.9181
NSGA-III	0.8103	0.2486	0.0298	0.5196	0.7831	0.2320	0.0288	0.5599
MOTAMA-GA-LS_1^b	0.7756	0.2206	0.0305	0.4698	0.7472	0.2231	0.0257	0.5296
MOTAMA-GA-LS_2^b	0.7721	0.2256	0.0375	0.4783	0.7366	0.2201	0.0318	0.5145
MOTAMA-AMDE-LS_1^c	0.6175	0.2986	0.0343	0.4548	0.5697	0.3375	0.0423	0.7012
MOTAMA-AMDE-LS_2^c	0.6094	0.3010	0.0411	**0.4498**	0.5595	0.3410	0.0431	0.7000
Instance	Real_1				Real_2			
MOTAMAQ	**0.9397**	**0.1009**	0.0252	0.6647	**0.9336**	**0.1127**	0.0278	0.5210
dε-MOEA	0.9163	0.1137	0.0246	0.6079	0.9063	0.1635	0.0268	0.6084
MOEA/D-DE	0.6640	0.2331	0.0737	1.0125	0.6017	0.2524	0.0667	0.9764
NSGA-III	0.8873	0.1198	0.0247	0.6518	0.8903	0.1322	0.0273	0.5430
MOTAMA-GA-LS_1^b	0.8900	0.1231	**0.0233**	0.6676	0.8867	0.1505	**0.0261**	0.5811
MOTAMA-GA-LS_2^b	0.9051	0.1179	0.0275	0.6561	0.9024	0.1347	0.0289	0.5487
MOTAMA-AMDE-LS_1^c	0.7552	0.1674	0.0457	**0.5983**	0.7393	0.1883	0.0545	0.5253
MOTAMA-AMDE-LS_2^c	0.7577	0.1695	0.0494	0.6158	0.7393	0.1920	0.0504	**0.5175**
Instance	Real_3							
MOTAMAQ	0.9209	**0.1035**	0.0205	0.5432				
dε-MOEA	**0.9429**	0.1141	0.0221	0.6095				
MOEA/D-DE	0.6472	0.2435	0.0542	0.9685				
NSGA-III	0.9187	0.1239	0.0202	0.5316				
MOTAMA-GA-LS_1^b	0.8859	0.1252	**0.0198**	0.6201				
MOTAMA-GA-LS_2^b	0.8894	0.1134	0.0296	0.5849				
MOTAMA-AMDE-LS_1^c	0.7488	0.1759	0.0440	**0.4877**				
MOTAMA-AMDE-LS_2^c	0.7370	0.1782	0.0477	0.4993				

Table 8

Statistical test results for comparing the eight approaches across rescheduling points on the 21 test instances (The sign of '+/−/=' in A vs. B indicates that according to the metric considered, algorithm A is significantly better than B, significantly worse than B, or there is no significant difference between A and B based on the Wilcoxon rank sum test with the significance level of 0.05).

Metrics		*HVR*	*IGD*	*Spacing*	*Spread*	*HVR*	*IGD*	*Spacing*	*Spread*
Instance		sT10_dT10_E5_SK4-5				sT10_dT10_E10_SK4-5			
MOTAMAQ vs. dε-MOEA	*p*-value	0.0224	0.0493	0.2089	0.7227	1.86E-6	0.1537	0.0519	0.7283
	sign	+	+	=	=	+	=	=	=
MOTAMAQ vs. MOEA/D-DE	*p*-value	0.0046	0.0045	0.0075	2.55E-9	7.39E-11	7.38E-10	2.84E-4	3.02E-11
	sign	+	+	+	+	+	+	+	+
MOTAMAQ vs. NSGA-III	*p*-value	0.1206	0.0836	0.3081	0.2108	2.06E-6	0.0288	0.1023	0.9234
	sign	=	=	=	=	+	+	=	=
MOTAMAQ vs. MOTAMA-GA-LS_1^b	*p*-value	8.12E-4	0.2198	0.9764	0.8130	3.99E-4	0.7394	0.2772	0.2519
	sign	+	=	=	=	+	=	=	=
MOTAMAQ vs. MOTAMA-GA-LS_2^b	*p*-value	0.0021	0.0668	0.7394	0.6574	1.34E-5	0.7394	0.6309	0.3403
	sign	+	=	=	=	+	=	=	=
MOTAMAQ vs.	*p*-value	8.56E-4	0.0444	0.1260	0.6789	3.69E-11	9.92E-7	0.9587	0.0224
	sign	+	+	=	=	+	+	=	+

MOTAMA-AMDE-LS$_1^c$									
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value sign	5.26E-4 +	0.0141 +	0.1188 =	0.7562 =	3.34E-11 +	1.21E-8 +	0.2458 =	0.0451 +
Instance		sT10_dT10_E15_SK4-5				sT10_dT10_E5_SK6-7			
MOTAMAQ vs. dε-MOEA	p-value sign	4.35E-5 +	4.22E-4 +	1.11E-6 +	0.1669 =	0.2009 =	0.0037 +	4.71E-4 +	0.3711 =
MOTAMAQ vs. MOEA/D-DE	p-value sign	3.02E-11 +	3.02E-11 +	3.82E-10 +	3.02E-11 +	5.57E-10 +	1.09E-10 +	6.51E-9 +	3.02E-11 +
MOTAMAQ vs. NSGA-III	p-value sign	4.06E-5 +	3.83E-4 +	0.1903 =	0.2308 =	6.62E-4 +	0.0040 +	1.32E-4 +	0.3001 =
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value sign	1.64E-5 +	0.2905 =	8.29E-6 +	0.6627 =	4.71E-4 +	0.0011 +	0.0030 +	0.2226 =
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value sign	4.22E-4 +	0.1087 =	2.92 E-9 +	0.0122 +	7.70E-4 +	0.0044 +	8.15E-5 +	0.9470 =
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value sign	7.38E-10 +	5.97E-9 +	8.48E-7 +	7.12E-9 +	4.99E-7 +	6.01E-6 +	1.17E-4 +	0.0215 −
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value sign	2.61E-10 +	1.31E-8 +	2.37E-7 +	5.00E-9 +	7.77E-7 +	1.20E-6 +	2.13E-5 +	0.0271 −
Instance		sT10_dT10_E10_SK6-7				sT10_dT10_E15_SK6-7			
MOTAMAQ vs. dε-MOEA	p-value sign	0.1373 =	0.0035 +	0.1858 =	0.4553 =	0.0099 +	9.21E-5 +	8.56E-4 −	0.4464 =
MOTAMAQ vs. MOEA/D-DE	p-value sign	7.12E-9 +	1.15E-7 +	6.01E-8 +	3.02E-11 +	3.02E-11 +	3.02E-11 +	1.61E-6 +	3.02E-11 +
MOTAMAQ vs. NSGA-III	p-value sign	0.2188 =	0.1203 =	0.2002 =	0.0116 +	0.0032 +	6.73E-5 +	7.83E-4 −	0.3111 =
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value sign	0.0067 +	2.83E-4 +	0.3329 =	0.7283 =	0.0170 +	0.2226 =	0.0519 =	0.2458 =
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value sign	0.0110 +	0.0451 +	0.9234 =	0.6952 =	0.0339 +	0.5201 =	0.6414 =	0.1055 =
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value sign	1.17E-4 +	4.44E-7 +	4.98E-4 +	0.0315 +	4.31E-8 +	1.21E-10 +	0.9470 =	0.1120 =
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value sign	3.18E-4 +	3.01E-7 +	2.39E-4 +	0.0261 +	5.09E-8 +	1.96E-10 +	0.3953 =	0.2707 =
Instance		sT20_dT10_E5_SK4-5				sT20_dT10_E10_SK4-5			
MOTAMAQ vs. dε-MOEA	p-value sign	0.3136 =	0.2833 =	0.4508 =	0.0811 =	0.0679 =	0.0023 +	0.0963 =	0.3478 =
MOTAMAQ vs. MOEA/D-DE	p-value sign	4.06E-10 +	7.25E-9 +	8.01E-6 +	5.31E-10 +	3.02E-11 +	3.02E-11 +	1.20E-8 +	3.02E-11 +
MOTAMAQ vs. NSGA-III	p-value sign	0.2743 =	0.0218 +	0.6682 =	0.0917 =	0.0213 +	0.0088 +	0.0878 =	0.7790 =
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value sign	0.1030 =	0.3121 =	0.9058 =	0.1433 =	0.0150 +	0.0207 +	0.0392 +	0.0484 +
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value sign	0.1360 =	0.2695 =	0.8825 =	0.8592 =	0.0594 =	0.0327 +	9.52E-4 +	0.6735 =
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value sign	1.18E-6 +	6.61E-9 +	0.3912 =	0.0604 =	3.65E-8 +	1.01E-8 +	0.3555 =	0.0049 −
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value sign	5.55E-7 +	2.13E-9 +	0.9906 =	0.0287 +	3.20E-9 +	6.01E-8 +	0.0099 +	0.0117 −
Instance		sT20_dT10_E15_SK4-5				sT20_dT10_E5_SK6-7			
MOTAMAQ vs. dε-MOEA	p-value sign	0.0307 +	0.0392 +	0.1669 =	0.9352 =	0.0421 +	0.2009 =	0.2707 =	0.7172 =
MOTAMAQ vs. MOEA/D-DE	p-value sign	3.02E-11 +	6.70E-11 +	0.0615 =	3.02E-11 +	3.34E-11 +	1.46E-10 +	0.0032 +	3.02E-11 +
MOTAMAQ vs. NSGA-III	p-value sign	2.08E-4 +	1.33E-4 +	0.1532 =	0.8702 =	0.0073 +	0.1760 =	0..2291 =	0.1906 =
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value sign	0.0051 +	0.0224 +	0.0392 −	0.6309 =	0.0019 +	0.4643 =	0.6735 =	0.1413 =
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value sign	0.0070 +	0.0424 +	0.1413 =	0.9000 =	0.0468 +	0.7506 =	0.3555 =	0.0824 =

MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value	1.20E-8	9.53E-7	0.0378	1.03E-6	7.70E-8	5.09E-8	0.2519	4.64E-5
	sign	+	+	+	+	+	+	=	+
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value	5.46E-9	1.03E-6	0.0905	1.73E-6	1.20E-8	1.70E-8	0.7845	1.78E-4
	sign	+	+	=	+	+	+	=	+
Instance		sT20_dT10_E10_SK6-7				sT20_dT10_E15_SK6-7			
MOTAMAQ vs. dε-MOEA	p-value	0.0260	4.57E-9	3.35E-8	4.74E-6	0.0401	9.52E-4	0.5395	0.6309
	sign	+	+	−	−	+	+	=	=
MOTAMAQ vs. MOEA/D-DE	p-value	3.02E-11	6.07E-11	0.1120	3.02E-11	3.02E-11	3.02E-11	1.86E-6	3.02E-11
	sign	+	+	=	+	+	+	+	+
MOTAMAQ vs. NSGA-III	p-value	0.0378	3.80E-4	3.17E-6	6.12E-6	0.0093	1.03E-5	0.7033	0.3174
	sign	+	+	−	−	+	+	=	=
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value	0.0051	0.0044	7.74E-6	3.65E-8	0.0122	0.0080	0.6309	0.2905
	sign	+	+	−	−	+	+	=	=
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value	0.0064	0.0076	0.0468	3.01E-7	1.49E-4	0.0451	0.0024	0.2170
	sign	+	+	−	−	+	+	+	=
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value	3.50E-9	2.49E-6	0.3329	0.1761	8.48E-9	3.96E-8	9.21E-5	3.83E-6
	sign	+	+	=	=	+	+	+	+
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value	6.53E-8	6.05E-7	0.5011	0.8534	1.70E-8	1.47E-7	6.91E-4	1.29E-6
	sign	+	+	=	=	+	+	+	+
Instance		sT30_dT10_E5_SK4-5				sT30_dT10_E10_SK4-5			
MOTAMAQ vs. dε-MOEA	p-value	0.2993	0.0011	0.0241	0.0287	0.5201	8.56E-4	0.9823	0.2282
	sign	=	+	+	+	=	+	=	=
MOTAMAQ vs. MOEA/D-DE	p-value	4.99E-9	4.68E-9	0.1023	3.66E-10	3.02E-11	3.02E-11	0.0091	3.02E-11
	sign	+	+	=	+	+	+	+	+
MOTAMAQ vs. NSGA-III	p-value	0.1846	0.2173	0.6930	0.1766	0.8902	0.1243	0.2208	0.2787
	sign	=	=	=	=	=	=	=	=
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value	0.0617	0.2413	0.2837	0.1983	0.0670	0.0635	0.3790	0.1809
	sign	=	=	=	=	=	=	=	=
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value	0.0720	0.9412	0.2282	0.0868	0.0043	0.0555	0.6309	0.7172
	sign	=	=	=	=	+	=	=	=
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value	1.80E-6	1.46E-5	0.9941	0.0434	2.39E-8	7.70E-8	0.7062	2.78E-7
	sign	+	+	=	=	+	+	=	+
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value	6.87E-7	1.28E-6	0.8302	0.0227	1.87E-7	1.87E-7	0.8303	5.19E-7
	sign	+	+	=	=	+	+	=	+
Instance		sT30_dT10_E15_SK4-5				sT30_dT10_E5_SK6-7			
MOTAMAQ vs. dε-MOEA	p-value	0.7618	0.0150	0.8534	0.8073	0.9470	0.0076	0.2226	0.0905
	sign	=	+	=	=	=	+	=	=
MOTAMAQ vs. MOEA/D-DE	p-value	5.49E-11	2.78E-7	1.11E-6	3.02E-11	3.02E-11	4.50E-11	0.2643	5.07E-10
	sign	+	+	+	+	+	+	=	+
MOTAMAQ vs. NSGA-III	p-value	0.0203	0.0026	0.0074	0.6109	0.0416	0.0033	0.0746	0.0892
	sign	+	+	+	=	+	+	=	=
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value	0.5692	0.1761	0.3632	0.3632	0.0327	0.1413	0.0963	0.0773
	sign	=	=	=	=	+	=	=	=
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value	0.0877	0.2398	0.9000	0.8650	0.0905	0.0242	0.0271	0.2282
	sign	=	=	=	=	=	+	+	=
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value	4.69E-8	1.73E-7	2.01E-4	2.60E-5	2.83E-8	1.56E-8	0.8534	0.1322
	sign	+	+	+	+	+	+	=	=
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value	5.53E-8	1.16E-7	2.13E-4	3.09E-6	1.17E-9	3.50E-9	0.2707	0.0017
	sign	+	+	+	+	+	+	=	+
Instance		sT30_dT10_E10_SK6-7				sT30_dT10_E15_SK6-7			
MOTAMAQ vs. dε-MOEA	p-value	0.3953	0.0044	1.04E-4	0.0993	0.0484	0.0436	0.0099	1.11E-6
	sign	=	+	−	=	+	+	+	+
MOTAMAQ vs. MOEA/D-DE	p-value	4.62E-10	7.38E-10	0.4918	3.02E-11	3.02E-11	6.70E-11	0.0292	3.02E-11
	sign	+	+	=	+	+	+	+	+
MOTAMAQ vs. NSGA-III	p-value	0.7026	0.0182	0.0081	0.1346	0.0311	0.0117	0.0196	8.36E-6
	sign	=	+	−	=	+	+	+	+
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value	0.0468	0.2226	0.0051	0.0023	0.0468	0.0453	0.8883	0.0850
	sign	+	=	−	−	+	+	=	=
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value	0.0112	0.1715	0.9587	0.0112	0.0016	0.0403	5.97E-5	0.6520
	sign	+	=	=	−	+	+	+	=
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value	6.53E-7	2.53E-4	0.0773	4.71E-4	3.82E-9	5.60E-7	1.85E-8	3.02E-11
	sign	+	+	=	−	+	+	+	+

MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value sign	1.60E-7 +	6.55E-4 +	0.6735 =	5.27E-5 −	2.61E-10 +	1.03E-6 +	4.44E-7 +	3.02E-11 +
Instance		Real_1				Real_2			
MOTAMAQ vs. dε-MOEA	p-value sign	0.0303 +	0.0468 +	0.1224 =	0.0053 −	0.0067 +	1.11E-4 +	0.5201 =	0.4740 =
MOTAMAQ vs. MOEA/D-DE	p-value sign	1.33E-10 +	3.20E-9 +	3.82E-10 +	8.99E-11 +	1.78E-10 +	5.97E-9 +	5.07E-10 +	3.02E-11 +
MOTAMAQ vs. NSGA-III	p-value sign	2.71E-4 +	0.0086 +	0.1266 =	0.1120 =	0.0086 +	0.0921 =	0.9722 =	0.0967 =
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value sign	6.91E-4 +	0.0029 +	0.2340 =	0.3017 =	7.70E-4 +	0.0624 =	0.9117 =	2.84E-4 +
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value sign	0.0163 +	0.2519 =	0.2398 =	0.1316 =	0.0025 +	0.0411 +	0.2707 =	0.0701 =
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value sign	1.55E-9 +	5.61E-5 +	0.0903 =	0.0026 −	3.35E-8 +	3.39E-5 +	2.23E-9 +	0.6952 =
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value sign	2.23E-9 +	2.43E-5 +	0.0619 =	0.0093 −=	8.89E-10 +	5.26E-5 +	2.67E-9 +	0.7731 =
Instance		Real_3							
MOTAMAQ vs. dε-MOEA	p-value sign	0.2282 =	0.0176 +	0.8766 =	0.0991 =				
MOTAMAQ vs. MOEA/D-DE	p-value sign	2.61E-10 +	3.96E-8 +	1.09E-10 +	3.34E-11 +				
MOTAMAQ vs. NSGA-III	p-value sign	0.3306 =	0.0161 +	0.9280 =	0.2679 =				
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	p-value sign	0.0067 +	0.0103 +	0.4553 =	0.0016 +				
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	p-value sign	0.0351 +	0.5592 =	8.15E-5 +	0.0378 +				
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	p-value sign	6.53E-8 +	3.77E-4 +	9.75E-10 +	0.0133 −				
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	p-value sign	1.25E-7 +	8.56E-4 +	1.07E-9 +	0.0451 −				

Table 9

Comparison results summarized from Table 8 (the percentage of the 18 random instances and 3 real-world instances for which the statistical tests indicate MOTAMAQ to be better, similar or worse than each of the seven other approaches)

Random Instances												
	HVR			*IGD*			*Spacing*			*Spread*		
MOTAMAQ vs. dε-MOEA	+	=	−	+	=	−	+	=	−	+	=	−
	50%	50%	0	83%	17%	0	22%	61%	17%	11%	83%	6%
MOTAMAQ vs. MOEA/D-DE	+	=	−	+	=	−	+	=	−	+	=	−
	100%	0	0	100%	0	0	72%	28%	0	100%	0	0
MOTAMAQ vs. NSGA-III	+	=	−	+	=	−	+	=	−	+	=	−
	67%	33%	0	72%	28%	0	17%	66%	17%	11%	83%	6%
MOTAMAQ vs. MOTAMA-GA-LS$_1^b$	+	=	−	+	=	−	+	=	−	+	=	−
	78%	22%	0	39%	61%	0	17%	66%	17%	6%	83%	11%
MOTAMAQ vs. MOTAMA-GA-LS$_2^b$	+	=	−	+	=	−	+	=	−	+	=	−
	72%	28%	0	44%	56%	0	33%	61%	6%	6%	83%	11%
MOTAMAQ vs. MOTAMA-AMDE-LS$_1^c$	+	=	−	+	=	−	+	=	−	+	=	−
	100%	0	0	100%	0	0	39%	61%	0	50%	33%	17%
MOTAMAQ vs. MOTAMA-AMDE-LS$_2^c$	+	=	−	+	=	−	+	=	−	+	=	−
	100%	0	0	100%	0	0	39%	61%	0	61%	22%	17%
Real-world Instances												
	HVR			*IGD*			*Spacing*			*Spread*		
MOTAMAQ vs. dε-MOEA	+	=	−	+	=	−	+	=	−	+	=	−
	67%	33%	0	100%	0	0	0	100%	0	0	67%	33%
MOTAMAQ vs. MOEA/D-DE	+	=	−	+	=	−	+	=	−	+	=	−
	100%	0	0	100%	0	0	100%	0	0	100%	0	0
MOTAMAQ vs. NSGA-III	+	=	−	+	=	−	+	=	−	+	=	−
	67%	33%	0	67%	33%	0	0	100%	0	0%	100%	0
MOTAMAQ vs.	+	=	−	+	=	−	+	=	−	+	=	−

MOTAMA-GA-LS_1^b	100%	0	0	67%	33%	0	0	100%	0	67%	33%	0
MOTAMAQ vs. MOTAMA-GA-LS_2^b	+	=	−	+	=	−	+	=	−	+	=	−
	100%	0	0	33%	67%	0	33%	67%	0	33%	67%	0
MOTAMAQ vs. MOTAMA-AMDE-LS_1^c	+	=	−	+	=	−	+	=	−	+	=	−
	100%	0	0	100%	0	0	67%	33%	0	0	33%	67%
MOTAMAQ vs. MOTAMA-AMDE-LS_2^c	+	=	−	+	=	−	+	=	−	+	=	−
	100%	0	0	100%	0	0	67%	33%	0	0	33%	67%

5.3.4 Comparison with the three state-of-the-art MOEAs

As explained in section 5.3.1, we have compared MOTAMAQ, dε-MOEA, MOEA/D-DE, and NSGA-III in order to evaluate the impact of different dynamic MOEAs on the performance of MODSPSP.

First, the convergence performance metrics *IGD* and *HVR* are considered. In terms of *IGD*, MOTAMAQ exhibits distinct advantages over dε-MOEA, MOEA/D-DE, and NSGA-III. It can be seen from Table 7 that the proposed approach (MOTAMAQ) has the best average value for 15 out of 18 random instances, and for all the real-world instances. From Table 9, it is clear that the performance of MOTAMAQ is significantly better than dε-MOEA and NSGA-III for 83% and 78% of the random instances, and for 100% and 67% of the real-world instances, respectively. In addition to that, MOTAMAQ outperforms MOEA/D-DE for all the random and real-world instances. These results clearly illustrate the ability of proposed approach in providing software managers with a more diverse set of non-dominated solutions that are closer to the reference Pareto front. Considering *HVR*, the performance of MOTAMAQ is better than or comparable to dε-MOEA and NSGA-III for all instances. Table 7 indicates that MOTAMAQ obtains the best average value for 11 out of the 18 random instances, and for 2 out of the 3 real-world instances. Table 9 shows that MOTAMAQ performs significantly better than dε-MOEA and NSGA-III for 50% and 67% of the random cases, and for 67% and 67% of the real-world instances, respectively. However, there is no significant difference between each pair of them for the remaining cases. One possible reason for this small gap is that dε-MOEA is good at maintaining a wide spread of solutions due to which its *HVR* value (which measures both the convergence and spread) increases. When compared with MOEA/D-DE, similar to *IGD,* MOTAMAQ outperforms MOEA/D-DE for all the random and real-world instances.

For the distribution performance *Spacing*, MOTAMAQ is comparable to dε-MOEA and NSGA-III, since there is no significant difference between them for 61% and 66% of the random instances, and for 100% and 100% of the real-world instances, respectively. For the random instances where there was a significant difference in terms of *Spacing*, MOTAMAQ was better than dε-MOEA and NSGA-III in around half and worse in around the other half of them. The performance of MOTAMAQ is significantly better than or comparable to that of MOEA/D-DE for all instances. In terms of *Spread*, the performance of MOTAMAQ is comparable to dε-MOEA for the random instances, while a bit worse than dε-MOEA for the real-world instances, which validates the assumption that dε-MOEA is able to find solutions with a good spread. There is no significant difference between MOTAMAQ and NSGA-III on 83% and 100% of the rand and real-world instances, respectively. Besides, MOTAMAQ performs significantly better than MOEA/D-DE for all the instances on *Spread*.

Since the convergence performance is the most important factor that a software manager should

consider when evaluating an approach, the improved convergence behavior of our MOTAMAQ approach over dε-MOEA, MOEA/D-DE, and NSGA-III clearly indicates that the strategies it adopts are very effective for solving MODSPSP. Such strategies include the Q-learning-based learning mechanism that chooses appropriate search operators to different environments; maintenance of two archives to promote convergence and diversity separately to handle many objectives; and incorporation of local search operators together with global search operators specifically designed for the MODSPSP. Considering its poor performances in terms of convergence, distribution and spread, MOEA/D-DE may not be suitable for solving MODSPSP.

5.3.5 Comparison with the four fixed actions

To further analyze the impact of the self-adaptive learning mechanism on the performance of our rescheduling approach, MOTAMAQ has been compared to MOTAMA-GA-LS$_1^b$, MOTAMA-GA-LS$_2^b$, MOTAMA-AMDE-LS$_1^c$, and MOTAMA-AMDE-LS$_2^c$. As stated in section 5.3.1, these four algorithms do not adopt the learning mechanism, but use the fixed search operators in different scheduling environments.

As mentioned before, in Table 7, MOTAMAQ has the best average *IGD* value for 15 out of the 18 random instances, and for all the real-world instances. The results shown in table 9 indicate that with *IGD*, MOTAMAQ is significantly better than MOTAMA-GA-LS$_1^b$ and MOTAMA-GA-LS$_2^b$ for 39% and 44% of the random cases, and for 67% and 33% of the real-world instances, respectively. For the remaining instances, no significant difference has been found. Besides, MOTAMAQ is significantly better than MOTAMA-AMDE-LS$_1^c$ and MOTAMA-AMDE-LS$_2^c$ for all the random and real-world instances. In terms of *HVR*, it can be seen from Table 7 that MOTAMAQ has obtained the best average value for 11 out of the 18 random instances, and for 2 out of the 3 real-world instances. Table 9 shows that for *HVR*, MOTAMAQ performs significantly better than MOTAMA-GA-LS$_1^b$ and MOTAMA-GA-LS$_2^b$ for 78% and 72% of the random cases, respectively and for all the three real-world instances. There is no significant difference between them for the remaining cases. Besides, the performance of MOTAMAQ is also significantly better than that of MOTAMA-AMDE-LS$_1^c$ and MOTAMA-AMDE-LS$_2^c$ for all the random and real-world instances. These results clearly indicate that the introduction of self-adaptive learning mechanism based on Q-learning acutely improves the convergence performance of MOEA-based rescheduling approach (especially for *HVR*). It means that the proposed approach MOTAMAQ has selected the right actions based on the learned information in most of the cases. Thus, when rescheduling, it is helpful for a software manager to take features of the current project environment into account, and choose an appropriate approach adaptively.

As for the metrics of *Spacing* and *Spread*, MOTAMAQ is comparable to MOTAMA-GA-LS$_1^b$ and MOTAMA-GA-LS$_2^b$, since there is no significant difference between each pair of them in most of the cases. Compared to MOTAMA-AMDE-LS$_1^c$ and MOTAMA-AMDE-LS$_2^c$, MOTAMAQ behaves better for *Spacing* since it outperforms or is comparable to them for all the instances. Meanwhile, considering the *Spread* metric, MOTAMAQ behaves a bit better for random instances since it is significantly better than each of them for 50% and 61% of the random instances, respectively. However, MOTAMA-AMDE-LS$_1^c$

and MOTAMA-AMDE-LS$_2^c$ have a better *Spread* performance as a whole for the real-world instances: they get the overall best average value on 2 and 1 out of the 3 real instances (see Table 7), respectively, and each of them is significantly better than MOTAMAQ for 2 out of the 3 real instances (see Tables 8 and 9).

5.4 Pareto fronts at rescheduling points

At each rescheduling point, a set of non-dominated solutions are evolved by MOTAMAQ. With the aim to demonstrate the trade-offs among these solutions that a software manager could exploit when making a selection about the final schedule, one rescheduling point on a test instance (sT20_dT10_E10_SK4-5) has been chosen arbitrarily and for the sake of illustration. At t_l=24.2, the employee e_4 returned, and a total of 7 tasks existed in the project. In order to show a five-objective Pareto front obtained by MOTAMAQ, a parallel coordinate plot is given in Fig. 9. Since the five objectives in MODSPSP have different scales, the best and the worst objective values observed during all the runs of the compared eight approaches have been identified, and the objective values of the obtained Pareto front are normalized. It can be seen from Fig. 9 that MOTAMAQ can find a set of well-distributed non-dominated solutions in the objective space, and the non-dominated front extends over a large range rather than being in a limited area. This result indicates the ability of MOTAMAQ in maintaining diversity.

To visually investigate different trade-offs among the five objectives obtained by MOTAMAQ, the diagonal plot [34] shown in Fig. 10 has been studied. The plot presents pairwise interactions among the five dimensions along the Pareto front, where the axes of any plot can be found by checking the corresponding diagonal boxes and their ranges. For example, the plot shown at the second row fifth column has its vertical axis as $cost_l$ and horizontal axis as *satisfaction*. First, it can be seen from the plot of $duration_l$ vs. $cost_l$ that the two efficiency objectives are conflicting with each other in most of the cases, since a smaller $duration_l$ often leads to a larger $cost_l$. Second, it can be observed from the plot of $duration_l$ vs. *satisfaction* that the two objectives are also conflicting with each other. When finding a solution that has a smaller $duration_l$, the *satisfaction* value becomes worse. Thirdly, the figure *stability* vs. *satisfaction* suggests that the two objectives have a similar variation tendency. When the *stability* value becomes smaller, the *satisfaction* value also decreases. However, it is hard to determine the relationship from the remaining figures. For instance, a small *robustness* may correspond to either a small or a high $cost_l$. The reason may be that a total of five objectives are optimized simultaneously, and the relationship between each pair of them becomes more complex, compared to the case in which only two or three objectives are considered together. No solution can simultaneously optimize all the considered objectives.

Some examples of the objective vectors chosen from the Pareto front are shown in Fig. 10 and listed in Table 10. A solution may behave very well for one objective, but poorly for some of the others, such as $Solution_1$ – $Solution_5$. Some solutions may obtain good (but not the best) values for most of the objectives, which shows a good trade-off among various objectives, such as $Solution_6$ – $Solution_8$. The Pareto front generated by MOTAMAQ can provide a software manager with a deeper insight into various compromises among many objectives. It is useful for him/her to make an informed decision about the best compromise according to his/her preference.

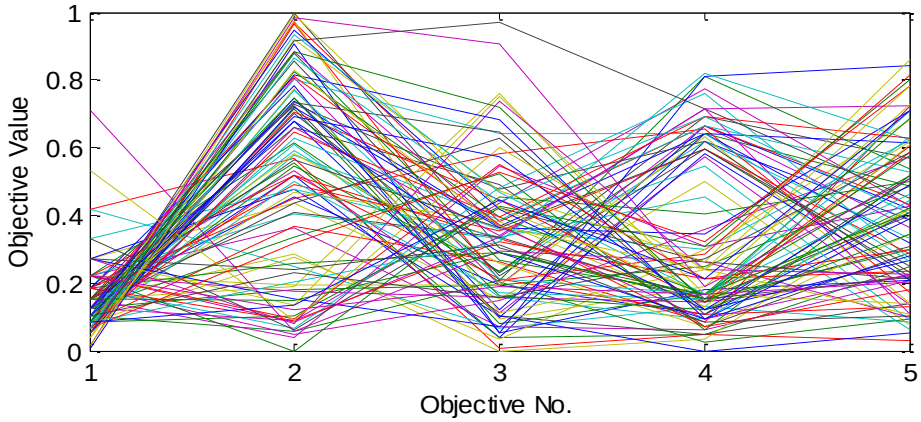

Fig. 9 Parallel coordinate plot of the Pareto front generated by MOTAMAQ at the rescheduling point $t_i = 24.2$ on sT20_dT10_E10_SK4-5.

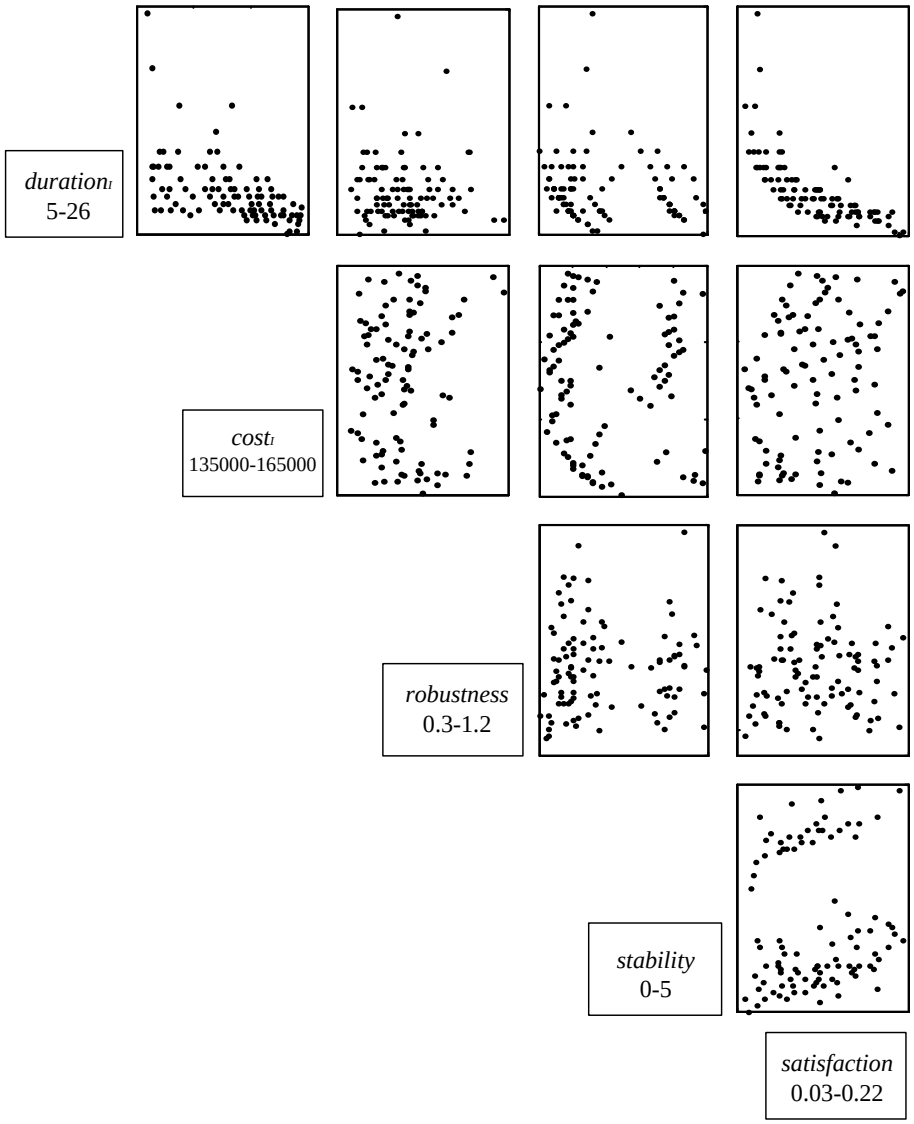

Fig. 10 Diagonal plot of the Pareto front generated by MOTAMAQ at the rescheduling point t_i=24.2 on sT20_dT10_E10_SK4-5.

Table 10

Examples of objective vectors chosen from the Pareto front generated by MOTAMAQ at t_f=24.2 on sT20_dT10_E10_SK4-5.

	[$duration_f$, $cost_f$, *robustness, stability*, *satisfaction*]
$Solution_1$	[5.01, 161291, 0.41, 4.86, 0.21]
$Solution_2$	[11.26, 135179, 0.74, 2.43, 0.14]
$Solution_3$	[9.21, 143452, 0.37, 0.21, 0.12]
$Solution_4$	[12.67, 148960, 0.45, 0, 0.043]
$Solution_5$	[16.89, 151524, 0.38, 0.29, 0.039]
$Solution_6$	[7.24, 145766, 0.59, 0.50, 0.18]
$Solution_7$	[8.44, 144839, 0.80, 0.36, 0.14]
$Solution_8$	[7.79, 148708, 0.69, 0.79, 0.16]

5.5 Discussion

When comparing different dynamic scheduling approaches, it is evident that the proposed Q-learning-based method MOTAMAQ outperforms the other three state-of-the-art MOEAs, named dε-MOEA, MOEA/D-DE, and NSGA-III. Despite the use of heuristic population initialization strategies, dε-MOEA MOEA/D-DE, and NSGA-III adopt fixed global search operators during the entire project implementation, without considering that different operators may be best suited for different environment states. In contrast, based on the states perceived by the agent in Q-learning and its accumulated knowledge, MOTAMAQ can capture the features of different scheduling environments, and then decide on the appropriate search operators. Thus, its adaptation to the changing environment is improved. Moreover, MOTAMAQ incorporates several problem specific local search operators to enhance the local search ability, and maintains a convergence archive and a diversity archive separately to handle many objectives. Experimental results in Section 5.3.4 show that these strategies combined promote the convergence of MOTAMAQ greatly, while maintaining a comparable distribution performance. In Section 5.3.5, comparisons with the four algorithms with fixed global and local search methods (i.e. one of the four actions in MOTAMAQ) further validate the effectiveness of our self-adaptive learning mechanism based on Q-learning. Besides, from the parallel coordinate plot and the diagonal plot of the Pareto front obtained by MOTAMAQ, a software manager can gain a deeper insight into the various trade-offs among the five objectives, and make an informed decision. The results above suggest that it is worthwhile employing MOTAMAQ as a dynamic scheduling approach to assist software project management.

6 Conclusions

This paper introduced a Q-learning-based multi-objective two-archive memetic algorithm to adapt to changing environments in dynamic software project scheduling. Our first contribution is to formulate a more practical formulation of the MODSPSP, which highlights the influence of human factors on project success. Our formulation relates the growth rate of the skill proficiency to both human factors (motivation, learning ability) and skill difficulties, being closer to the real world than other existing SPSP formulations. Considering the degree with which each employee is willing to engage with each skill (and thus with each task), the objective of employees' satisfaction is defined and considered together with project duration, cost,

robustness and stability under a variety of practical constraints at each rescheduling point.

Our second contribution is the design of a Q-learning-based multi-objective two-archive memetic algorithm to solve the formulated MODSPSP in a proactive-rescheduling way. The approach introduces problem specific local search operators to enhance the local search ability. It can perceive the state of the current project environment, and learn the appropriate global and local search operators of the memetic algorithm adaptively, according to the obtained information and the knowledge accumulated by the Q-learning agent. Besides, to deal with many objectives, it maintains two archives that promote convergence based on the $I_{\varepsilon+}$ indicator, and keep diversity based on Pareto dominance separately.

Our third contribution is a comprehensive experimental study of the newly proposed approach MOTAMAQ. The study is divided into three groups. The first group compares the overall performance in dynamic environments produced by MOTAMAQ and three state-of-the-art MOEA-based rescheduling methods, namely dε-MOEA, MOEA/D-DE, and NSGA-III. Experimental results show that the strategies designed in MOTAMAQ are very effective in improving its convergence performance. The Q-learning-based learning mechanism can adapt appropriate search operators to different scheduling environments. By cooperating with the global search operators, the problem-specific local search operators enhance the local search ability of the algorithm. Besides, the maintenance of two archives that promote convergence and diversity separately can deal with the 5 objectives in MODSPSP effectively. The second group compares MOTAMAQ with four algorithms which use fixed global and local search operators in different environments. Our results further demonstrate that the introduction of self-adaptive learning mechanism based on Q-learning helps to improve the convergence performance of our MOEA-based rescheduling approach. It indicates that MOTAMAQ has selected the right actions according to the learned information in most of the cases. The third group analyses different trade-offs among the five objectives. The parallel coordinate plot shows that MOTAMAQ can find a set of well-distributed non-dominated solutions in the objective space, and the non-dominated front extends over a large range rather than being in a limited area. The diagonal plot presents pairwise trade-offs among the five objectives along the Pareto front, from which a software manager can get a deeper insight into various compromises among many objectives, and make an informed decision.

Although our MODSPSP model is an improvement and considers more aspects of reality than the existing models (e.g., employees' subjective properties such as motivation and willingness to engage in tasks involving certain skills), it is still far from extracting all factors, uncertainties and dynamic events which could affect project scheduling environments. In our current work, the learning ability factor and the motivation factor of each employee are assumed to follow the normal distribution. We believe that some experiments need to be realized to investigate the suitability of normal distribution in modelling such factors while considering special situations such as more specialized subjects in which employees should have high motivation and the fact that their specialization is directly connected to the ability to learn new things. This would require data collection on the learning ability factor and the motivation factor of each employee gathered during a period of time. When the appropriate probability distribution is found, it can be easily incorporated in our method. The same holds for the deviations of task effort estimations. Besides,

more efficient global and local search operators could be designed, which could provide a diverse set of candidate actions for the agent to select, and further improve the search efficiency of our approach.

As future work, first, a more sophisticated mathematical formulation of the MODSPSP should be created. Different skills may require different specializations and learning abilities. Thus, if specialties of the employees who are engaged in the skills match well with the skill requirements, the project efficiency can be improved to a large extent. In addition, more properties of employees and tasks in the real project scheduling situations, e.g. employees' experiences, due-date of each task, as well as the elements that can affect the properties, e.g. political behaviors, social or human capital, psychological factors, should be taken into account. Meanwhile, how to measure these properties and factors remains a challenging study. Second, the proposed Q-learning-based dynamic scheduling approach MOTAMAQ should be applied to more complex software projects, with different kinds of dynamic events and uncertainties, e.g. changing objectives to be optimized, task removal, variations in the task precedence, and changes of the degree with which each employee is willing to engage with a skill. The relationship between such dynamic factors and the performance of MOTAMAQ needs to be studied. Third, the practicability of the proposed scheduling approach should be further improved in terms of how close it is to real software project scenarios. This can be supported by performing a thorough empirical validation in a variety of industrial contexts, collecting large amounts of data from real-world projects, getting feedback from software developers on the assumptions made by our approach and on how to improve our method. Finally, the scalability of MOTAMAQ should also be validated, by applying it for scheduling larger scale software projects with a greater number of tasks and employees.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61502239, No. 61573361, and No. 61503191, and Natural Science Foundation of Jiangsu Province of China under Grant No. BK20150924 and No. BK20150933. We are grateful to Weineng Chen and Jun Zhang for providing the data of the three real-world PSP instances.

References

[1] E. Alba, J.F. Chicano, Software project management with Gas, Information Sciences, 177 (2007) 2380-2401.

[2] C.K. Chang, M.J. Christensen, T. Zhang, Genetic algorithms for project management, Annals of Software Engineering, 11 (2001) 107-139.

[3] C.K. Chang, H. Jiang, Y. Di, D. Zhu, Y. Ge, Time-line based model for software project scheduling with genetic algorithms, Information and Software Technology, 50 (2008) 1142-1154.

[4] W.N. Chen, J. Zhang, Ant colony optimization for software project scheduling and staffing with an event-based scheduler, IEEE Transactions on Software Engineering, 39(1) (2013) 1-17.

[5] F. Chicano, F. Luna, A.J. Nebro, E. Alba, Using multiobjective metaheuristics to solve the software project scheduling problem, in: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference, 2011, pp. 1915-1922.

[6] J.W. Christopher, D. Peter, Q-learning, Machine Learning, 8 (1992) 279-292.

[7] B. Crawford, R. Soto, F. Johnson, E. Monfroy, F. Paredes, A max–min ant system algorithm to solve the software project scheduling problem, Expert Systems with Applications, 41 (2014) 6634-6645.

[8] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints

[9] K. Deb, M. Mohan, S. Mishra, Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal, Evolutionary Computation, 13(4) (2005) 501-525.

[10] K. Deb, R.B. Agrawal, Simulated binary crossover for continuous search space, Complex Systems, 9(2) (1995) 115~148.

[11] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6 (2) (2002) 182-197.

[12] J. Deng, L. Wang, A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem, Swarm and Evolutionary Computation, 32 (2007) 121-131.

[13] A.E. Eiben, J.E. Smith, Introduction to evolutionary computing, Springer-Verlag, Berlin, 2003.

[14] AP. Engelbrecht, G. Pampara, Binary differential evolution strategies, in: Proceedings of IEEE Congress on Evolutionary Computation, Singapore, 2007, pp. 1942-1947.

[15] D. Gong, N. Qin, X. Sun, Evolutionary algorithms for optimization problems with uncertainties and hybrid indices, Information Sciences, 181 (19) (2011) 4124-4138.

[16] S. Gueorguiev, M. Harman, G. Antoniol, Software project panning for robustness and completion time in the presence of uncertainty using multi-objective search based software engineering, in: Proceedings of 11th Annual Genetic and Evolutionary Computation Conference, 2009, pp. 1673-1680.

[17] M. Hapke, A. Jaszkiewicz, R. Slowinski, Fuzzy project scheduling system for software development, Fuzzy Sets and Systems, 67(1)(1994) 101-117.

[18] M. Harman, The current state and future of search based software engineering, In: Proceedings of the 2007 Future of Software Engineering, ACM, 2007, pp. 342-357.

[19] K. Hwang, H. Lin, Y. Hsu, H. Yu, Self-organizing state aggregation for architecture design of Q-learning, Information Sciences, 181 (2011) 2813-2822.

[20] Intaver Institute Inc, Software project scheduling under uncertainties, <http://www.intaver.com/Articles/Article_Software Project Management.pdf>.

[21] E. Kocaguneli, T. Menzies, A.B. Bener, J.W. Keung, Exploiting the essential assumptions of analogy-based effort estimation, IEEE Transactions on Softwre Engineering, 38(2) (2012) 425-438.

[22] S. Lazarova-Molnar, R. Mizouni, A simulation-based approach to enhancing project schedules by the inclusion of remedial action secnarios, in: Proceedings of the 2011 Winter Simulation Conference, 2011, pp. 761-772.

[23] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Transactions on Evolutionary Computation, 13(2) (2009) 284-302.

[24] F. Luna, D. González-Álvarez, F. Chicano, M.A. Vega-Rodríguez, The software project scheduling problem: a scalability analysis of multi-objective metaheuristics, Applied Soft Computing, 15 (2014) 136-148.

[25] Y. Mei, K. Tang, X. Yao, Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem, IEEE Transaction on Evolutionary Computation, 15(2) (2011) 151-165.

[26] E. Mezura-Montes, M. Reyes-Sierra, C.A.C. Coello, Multi-objective optimization using differential evolution: a survey of the state-of-the-art, Advances in Differential Evolution, 2008, 173-196.

[27] L.L. Minku, D. Sudholt, X. Yao, Improved evolutionary algorithm design for the project scheduling problem based on runtime analysis, IEEE Transactions on Software Engineering, 40(1) (2014) 83-102.

[28] P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: toward memetic algorithms, Technical

Report ,Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, 1989.

[29] N. Nan, D.E. Harter, Impact of budget and schedule pressure on software development cycle time and effort, IEEE Transaction on Software Engineering, 35(5) (2009) 624-637.

[30] Q.K. Pan, L. Wang, H.Y. Sang, J.Q. Li, M. Liu, A high performing memetic algorithm for the flowshop scheduling problem with blocking, IEEE Transactions on Automation Science and Engineering, 10(3) (2013) 741-756.

[31] R.S. Pressman, software engineering: a practitioner's approach, sixth ed.. McGraw-Hill Science, 2005.

[32] Y. Sakurai, K. Takada, T. Kawabe, S. Tsuruta, A method to control parameters of evolutionary algorithms by using reinforcement learning, in: Proceedings of the 2010 Sixth International Conference on Signal-Image Technology and Internet Based Systems, IEEE, 2010, pp. 74-79.

[33] J.R. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization, Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995.

[34] R. Shang, L. Jiao, F. Liu, W. Ma, A novel immune clonal algorithm for MO problems. IEEE Transaction on Evolutionary Computation, 16(1) (2012) 35-50.

[35] X.N. Shen, L.L. Minku, R. Bahsoon, X. Yao, Dynamic Software Project Scheduling through a Proactive-Rescheduling Method, IEEE Transaction on Software Engineering, 42(7) (2016) 658-686.

[36] I. Sommerville, Software engineering, eighth ed.. Essex: Addison-Wesley, 2006.

[37] R.S. Sutton, A.G. Bart, Reinforcement learning: An introduction, The MIT Press, Cambridge, MA, 1998.

[38] D.A. Van Veldhuizen, G.B. Lamont, Multiobjective evolutionary algorithm test suites, in: Proceedings of 1999 ACM Symposium on Applied Computing, 1999, pp. 351-357.

[39] P. Vasant , G. Weber, V. N. Dieu. Handbook of research on modern optimization algorithms and applications in engineering and economics. Hershey, PA: IGI Global. 2016.

[40] H. Wang, L. Jiao, X. Yao, Two_Arch2: an improved two-archive algorithm for many-objective optimization, IEEE Transaction on Evolutionary Computation, 19(4) (2015) 524-541.

[41] H. Wang, X. Wang, X. Hu, X. Zhang, M. Gu. A multi-agent reinforcement learning approach to dynamic service composition, Information Sciences, 363 (2016) 96-119.

[42] X. Wu, P. Consoli, L.L. Minku, G. Ochoa, X. Yao. An evolutionary hyper-heuristic for the software project scheduling problem, in: Proceedings of the 14th International Conference on Parallel Problem Solving from Nature (PPSN'16), 2016, pp. 37-47.

[43] J. Xiao, L.J. Osterweil, Q. Wang, M. Li, Dynamic resource scheduling in disruption-prone software development environments, in: Proceedings of 13th International Conference on Fundamental Approaches to Software Engineering, 2010, pp. 107-122.

[44] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation, 11(6) (2007) 712-731.

[45] Z. Zhu, J. Xiao, S. He, Z. Jia, Y. Sun. A multi-objective memetic algorithm based on locality-sensitive hashing for one-to-many-to-one dynamic pickup-and-delivery problem, Information Sciences, 329 (2016) 73-89.

[46] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: Proceedings of Parallel Problem Solving from Nature—PPSN VIII, Springer Berlin, Germany, 2004, pp. 832-842.

[47] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation, 3(4) (1999) 257-271.