Expert Systems with Applications 41 (2014) 1830–1846

Contents lists available at ScienceDirect

Expert Systems with Applications

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

A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems

Madjid Tavana [a,b,*], Amir-Reza Abtahi [c], Kaveh Khalili-Damghani [d]

[a] Business Systems and Analytics Department, Lindback Distinguished Chair of Information Systems and Decision Sciences, La Salle University, Philadelphia, PA 19141, USA
[b] Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, D-33098 Paderborn, Germany
[c] Department of Knowledge Engineering and Decision Sciences, University of Economic Sciences, Tehran, Iran
[d] Department of Industrial Engineering, South-Tehran Branch, Islamic Azad University, Tehran, Iran

ARTICLE INFO

Keywords:
Project scheduling problem
Discrete time–cost trade-off problem
Multi-objective evolutionary algorithm
Activity preemption
Generalized precedence relations

ABSTRACT

Considering the trade-offs between conflicting objectives in project scheduling problems (PSPs) is a difficult task. We propose a new multi-objective multi-mode model for solving discrete time–cost–quality trade-off problems (DTCQTPs) with preemption and generalized precedence relations. The proposed model has three unique features: (1) preemption of activities (with some restrictions as a minimum time before the first interruption, a maximum number of interruptions for each activity, and a maximum time between interruption and restarting); (2) simultaneous optimization of conflicting objectives (i.e., time, cost, and quality); and (3) generalized precedence relations between activities. These assumptions are often consistent with real-life projects. A customized, dynamic, and self-adaptive version of a multi-objective evolutionary algorithm is proposed to solve the scheduling problem. The proposed multi-objective evolutionary algorithm is compared with an efficient multi-objective mathematical programming technique known as the efficient ε-constraint method. The comparison is based on a number of performance metrics commonly used in multi-objective optimization. The results show the relative dominance of the proposed multi-objective evolutionary algorithm over the ε-constraint method.

1. Introduction

Project scheduling problems (PSPs) have received significant attention and played a vital role in managing organizational resources. A PSP is defined by its activities (each with a specific execution time) and by the precedence relations among them. The overall goal in project scheduling is to optimize a set of measurement functions subject to a set of precedence and resource constraints (Singh & Ernst, 2011). PSPs are some of the most intractable problems in operations research, and have therefore become a popular playground for the latest optimization techniques (Baptiste & Demassey, 2004; Möhring, Schulz, Stork, & Uetz, 2003). PSPs are complex scheduling problems with limited resources. This extension of the PSP is called the resource-constrained PSP (RCPSP). Two types of constraints are usually present in a RCPSP: precedence constraints and resource constraints. Precedence constraints establish a specific sequence for some pairs of activities and resource constraints model the resource requirements of the activities assuming a limited resource supply (Correia, Lourenço, & Saldanha-da-Gama, 2012). Evolutionary optimization approaches such as particle swarm optimization (PSO) and genetic algorithms (GA) have been successfully used to solve the RCPSPs (Chen, 2011; Chen, Wu, Wang, & Lo, 2010; Hartmann, 2002; Van Peteghem & Vanhoucke, 2010.

In project scheduling, it is often possible to reduce the duration of some activities and thereby expedite the project duration with some additional costs. Project expedition decisions has traditionally involved time and cost trade-off considerations. However, it was recently suggested that the quality of a project should also be taken into consideration (Iranmanesh, Skandari, & Allahverdiloo, 2008; Tareghian & Taheri, 2006).

In the continuous trade-off problems, there are functions which correlate the time, cost, and quality objectives. As research efforts progressed in the field and practical needs arose, researchers began to focus on the development of procedures for solving the discrete time–cost trade-off problems (DTCTPs) (Hazır, Erel, & Günalay, 2011; Sonmez & Bettemir, 2012; Wuliang & Chengen, 2009; Xu, Zheng, Zeng, Wu, & Shen, 2012). In the discrete variant, the relationships between the objectives in a project are defined at discrete points. In this case, each activity can be executed in several

* Corresponding author at: Business Systems and Analytics Department, Lindback Distinguished Chair of Information Systems and Decision Sciences, La Salle University, Philadelphia, PA 19141, USA. Tel.: +1 215 951 1129; fax: +1 267 295 2854.
E-mail addresses: tavana@lasalle.edu (M. Tavana), amir_abtahi@yahoo.com (A.-R. Abtahi), kaveh.khalili@gmail.com (K. Khalili-Damghani).
URLs: http://tavana.us/ (M. Tavana), http://www.kaveh-khalili.webs.com (K. Khalili-Damghani).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.081

modes. Hence, the feasible solution space of the problem exponentially increases for medium and large size problems. These trade-off problems are known as non-deterministic polynomial-time hard (NP-Hard) (De, Dunne, Ghosh, & Wells, 1997).

DTCTP is inherently difficult to solve (Tareghian & Taheri, 2007). Prabudha, Dunne, Ghosh, and Wells (1995) have offered two reasons why interest in DTCTP should be revived. "First, discrete alternatives are common in practice and second discretization provides a convenient means for modeling any general time/cost relationship". Several exact and approximation procedures are developed for small-size trade-off problems (Burns, Liu, & Feng, 1996; Demeulemeester, De Reyck, & Herroelen, 2000; Skutella, 1998; Sunde & Lichtenberg, 1995). The solution procedures to the DTCTPs are classified into three groups: (a) Exact algorithms, such as linear programming, integer programming, dynamic programming, branch-and-bound algorithms, etc. (Erenguc, Ahn, & Conway, 2001), (b) Heuristic algorithms (Vanhoucke, Debels, & Sched, 2007), and (c) Meta-heuristic algorithms (Afshar, Kaveh, & Shoghli, 2007; Azaron, Perkgoz, & Sakawa, 2005; Chao-guang, Shang, Yan, Yuan-min, & Zhen-dong, 2005; El-Rayes & Kandil, 2005; Wuliang & Chengen, 2009; Yang, 2011; Zhang & Xing, 2010).

Szmerekovsky and Venkateshanb (2012) proposed four integer programming formulations for the irregular costs PSP with time–cost trade-offs. Three formulations using the standard assignment type variables were tested against a more novel integer programming formulation. Their empirical tests showed that in many instances the new formulation performed best and could solve problems with up to 90 activities in a reasonable amount of time.

Heuristic and Meta-heuristic algorithms represented better results for medium and large-size trade-off problems (Rahimi & Iranmanesh, 2008). Meta-heuristics have also been successfully used to solve multi-objective trade-off problems. Other assumptions such as time-switch constraints have been introduced in the literature on the trade-off problems (Vanhoucke, 2005). Table 1 represents some relevant studies on the trade-off problems in the literature.

Real world tasks, including project activities and job scheduling, can be either preemptable or non-preemptable (Błażewicz, Ecker, Pesch, Schmidt, & Węglarz, 2007). An activity is preemptable if it can be preempted at any time and restarted later with no cost (Demeulemeester & Herroelen, 2002). Preemption may be either discrete or continuous. In discrete case, preemption is allowed at the end of the time period while in continuous preemption, activity preemption may occur at an arbitrary time instant. Considering the assumption that preemptive activities can be preempted at integer time instants and restarted later at no additional cost, an activity is split into a number of sub-activities with unit duration. This type of preemption was first introduced by Kaplan (1988) in preemptive RCPSPs. Kaplan (1988) used dynamic programming to formulate the preemptive RCPSP and showed that this type of preemption has no meaningful effect on project lengths when constant resource availability levels are defined and the exact procedures are used. Demeulemeester and Herroelen (1996) later improved the formulation of Kaplan (1988) through a branch-and-bound procedure. Preemption may occur subject to a maximum limitation. Although allowing activity interruption may reduce the duration of a project, the repeated stopping and starting of an activity may not be feasible in practice (Ballestín, Valls, & Quintanilla, 2008).

Lino (1997) conducted an extensive set of experiments on scheduling of projects with generalized precedence relations and no resource constraints. Lino (1997) considered three different assumptions for modeling preemption as follows: (a) no interruption, (b) any number of interruptions at integer time instants, and (c) a maximum of one interruption per activity. Lino (1997) used an extensive number of randomly generated instances and showed that if each activity is allowed to be interrupted just once, then a significant reduction in project length is obtained in comparison with the case of no interruption. He also detected that allowing more than one interruption instead of a maximum of one interruption per activity does not further reduce the project length in the majority of the instances – and that when a reduction happens it is very small.

In recent years, the applications of the RCPSP and its extensions have attracted increasing interest from researchers and practitioners (Demeulemeester & Herroelen, 2002). One of the extensions of the RCPSP that has received considerable attention has been the RCPSP with preemption. Yang and Chen (2000) investigated one type of time constraint called a time-switch constraint which assumes that an activity begins at a specific time interval in a cycle with some pairs of exclusive components. Yang and Chen (2000) developed polynomial time algorithms to find the longest path (or critical path) and analyzed the float of each arc in this time-constrained activity network. Their analysis showed that the critical path and float in the time-constrained activity networks differ from those of the traditional activity networks and the consideration of the time-switch constraints lead to effective use of budgets and resources.

Table 1
A summary of the studies on the TCQT problem.

Author(s)	Method	Main contributions
Babu and Suresh (1996)	Linear programming	Using three inter-related linear programming models and extending them into non-linear models
Khang and Myint (1999)	Linear programming	Applying the Babu and Suresh (1996) method to an actual cement factory and examining the method's applicability, assumptions and limitations
El-Rayes and Kandil (2005)	Genetic algorithm	Applying their model to highway construction projects; quantifying quality with some quality indices and calculating the project quality based on an additive weighting method
Tareghian and Taheri (2006)	Integer programming	Developing a method to prune the activity execution modes
Pollack-Johnson and Liberatore (2006)	Goal programming	Conceptualizing the quality in projects; Quantifying the quality value of each activity execution mode with the Analytic Hierarchy Process (AHP) and developing a goal programming model with four objectives including: time, cost, minimum quality and mean of quality
Tareghian and Taheri (2007)	Electromagnetic scatter search	Validating and checking the applicability of their algorithm by solving a randomly generated large-scale problem with 19900 activities
Afshar et al. (2007)	Multi-colony ant algorithm	Solving an example and comparing their algorithm's results with some other algorithms
Zhang and Xing (2010)	Particle swarm optimization	Considering construction methods instead of execution modes for each activity; Using fuzzy numbers to describe time, cost, and quality; Using fuzzy multi-attribute utility methodology and constrained fuzzy arithmetic operators to evaluate each construction method; Demonstrating the effectiveness of their algorithm by solving a bridge construction problem
Kim, Kang, and Hwang (2012)	Mixed integer linear programming	Focusing on minimizing quality loss cost instead of maximizing the individual activity quality of the projects; Validating their model by applying it to a robot type palletizing system installation project

Vanhoucke, Demeulemeester, and Herroelen (2002) further studied the time-switch constraints and showed that these constraints can be incorporated into the well-known discrete time/cost trade-off problems to cope with day, night and weekend shifts. Vanhoucke (2005) showed that the time-switch constraints proposed by Yang and Chen (2000) impose a specific starting time on the project activities and force them to be inactive during the specified time periods. Vanhoucke (2005) proposed a new branch-and-bound algorithm which made use of a lower-bound calculation for the discrete time/cost trade-off problem.

Ballestín et al. (2008) widely studied the discrete activity preemption and showed how three basic elements of many heuristics for the RCPSP (i.e., codification, serial Schedule Generation Schema (SGS), and double justification) can be adapted to deal with interruption. They focused on a generalization of the RCPSP where a maximum of one interruption per activity is allowed. They also showed how these three elements can be further adapted to deal with more general preemptive problems. Ballestín, Valls, and Quintanilla (2009) further investigated the effect of interruption on project length in more general cases and proposed a new model and a Meta-heuristic algorithm that covered most practical applications of discrete activity preemption. Ballestín et al. (2009) also analyzed the usefulness of preemption in the presence of due dates.

In this paper a new multi-objective multi-mode preemptive generalized precedence discrete time–cost–quality trade-off (PGP-DTCQT) problem is proposed. The activities in the proposed PGP-DTCQT model are assumed to have several execution modes with specified time–cost–quality in each mode. The proposed PGP-DTCQT model has several features including:

- Three conflicting objectives including time, cost, and quality are considered.
- Specific upper and lower-bounds are assumed for the time, cost, and quality attributes of the project.
- The activities can change their execution mode after preemption.
- A maximum number of preemptions are allowed for each activity.
- A predetermined maximum time is assumed between two sequential preemptions.
- A predefined required minimum run time without any preemption is considered for each activity.
- Generalized Precedence relation is assumed among the activities of project.

These assumptions are often consistent with real-life projects. To our best knowledge, there is no trade-off model in the literature which considers all the aforementioned properties simultaneously. The PGP-DTCQT problem is modeled using a new multi-objective mixed-integer mathematical formulation. Two different procedures, including an efficient version of the ε-constraint method and a dynamic self-adaptive version of a multi-objective evolutionary computation are proposed to solve and generate sets of non-dominated solutions on the Pareto front of several instances of the PGP-DTCQT problem. The performance of both methods is compared on different sets of systematically generated instances of the PGP-DTCQT problem through different diversity and accuracy metrics in a multi-objective environment.

The remainder of the paper is organized as follows. In Section 2, we present our notations and problem formulation. In Section 3, we provide a comprehensive detail of our solution procedures. In Section 4, we present our computational experiments and results. In Section 5, we conclude with our conclusions and future research directions.

2. Notations and problem formulation

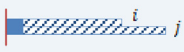
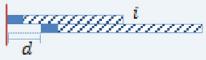
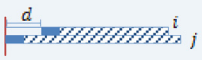
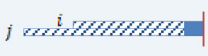
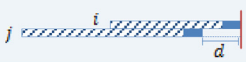
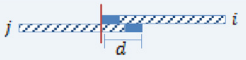
A project consists of $n+2$ activities numbered from 0 to $n+1$ where dummy activities $i=0$ and $i=n+1$ represent the beginning and the end of the project. Each project can be defined as an acyclic directed graph $G=(V,E)$, where V denotes the set of nodes representing *activities* and E denotes the set of arcs representing *relations* in the activity on node (AON) representation. Based on the preemption feature, the set V is divided into two distinct sets V_1 and V_2 which are respectively associated with the preemptive and non-preemptive activities. It is clear that $V=V_1\cup V_2$ and $\varnothing=V_1\cap V_2$. Each activity $i\in V_L,\ L=1,2$ can be executed in $r(i)\in k$ different modes where mode k has its own time (d_{ik}), cost (c_{ik}) and quality (q_{ik}). Dummy activities $i=0$ and $i=n+1$ have zero duration, cost and quality. There are also generalized precedence relations between the activities.

2.1. Generalized precedence relations

The generalized precedence relations are temporal constraints in which the starting or finishing times of a pair of activities have to be separated by at least or at most an amount of time denoted as "time lag" (Bianco & Caramia, 2012). The generalized precedence relations can be classified into "start-to-start" (SS), "finish-to-finish" (FF), "start-to-finish" (SF), "finish-to-start" (FS) relations (Crandall, 1973). These generalized precedence relationships are depicted in Table 2.

To consider generalized precedence constraints in the modeling procedure, the set of arcs, E, is classified into the following four subsets: E_{SS}, E_{SF}, E_{FS}, and E_{FF}, which respectively denote the sets

Table 2
The generalized precedence relations.

Precedence Relations	Time Lag		
	Zero	Positive	Negative
FS			
SS			
FF			
SF			

of start-to-start, start-to-finish, finish-to-start, and finish-to-finish precedence relations.

2.2. Preemption assumptions

Preemption is generally a situation in which an activity in the project can be stopped at a certain point in one phase of the planning process and continued again from that point in another phase of the planning process. A preempted activity uses no resources and those unused resources could be allocated to other activities. Preemption often occurs as a result of resource restrictions, unplanned quality control problems, customer service issues, or technical problems, among others. Several preemption assumptions are considered in project planning and control. An activity can be preempted in each phase of the planning process. The number of activity preemptions is restricted to eliminate the potential risk of restarting the activity from the beginning. In addition, the time period between the preemption and the re-starting points should be limited to avoid the risk of restarting the activity. Sometimes it may be possible to re-start an activity form the preemption point with a different execution mode due to technological or contractual changes in a project. We consider all aforementioned common project management assumptions with regard to preemption in the proposed model.

More specifically, denoting the maximum number of allowed preemption for activity i as $Maxnpre_i$, the maximum number of parts (sub-activities) of an activity in any feasible solution of the trade-off problem can be defined as $Maxnsub_i = Maxnpre_i + 1$. For each activity i, a minimum execution time of ε_{ik} is defined during which activity i in mode k has to be in progress without interruption. For each activity i, a maximum interruption time of α_i is defined. After each interruption, activity i has to be re-scheduled in a time period less than or equal to α_i. Otherwise, it is assumed to be a re-work and must be re-started. Consequently, the cost of the activity will increase and the quality will be diminished considerably. After each preemption activity i can be re-started in a different execution mode. Therefore, parts of a given preemptive activity may be executed in several modes.

2.3. Modeling the PGP-DTCQT

The multiple objectives in the PGP-DTCQT model are to concurrently minimize the overall time and cost and maximize the overall quality of the project. The notations used to formulate the PGP-DTCQT model are presented in Table 3.

The following multi-objective mixed integer non-linear mathematical programming formulation is proposed for the PGP-DTCQT model:

$$\text{Min } Time = S_{n+1,0} \tag{1}$$

$$\text{Min } Cost = \sum_{\substack{i=1\\ i\in V_2}}^{n}\sum_{k=1}^{r(i)} x_{ik}c_{ik} + \sum_{p=1}^{Maxnsub_i}\sum_{\substack{i=1\\ i\in V_1}}^{n}\sum_{k=1}^{r(i)} x_{ik,p}c_{ik,p} \tag{2}$$

$$\text{Max } Quality = \sum_{\substack{i=1\\ i\in V_2}}^{n}\sum_{k=1}^{r(i)} x_{ik}q_{ik} + \sum_{p=1}^{Maxnsub_i}\sum_{\substack{i=1\\ i\in V_1}}^{n}\sum_{k=1}^{r(i)x_{ik,p}q_{ik,p}} \tag{3}$$

s.t.

$$\sum_{k=1}^{r(i)} x_{ik} = 1 \quad \forall i \in V_2 \tag{4}$$

$$\sum_{k=1}^{r(i)} x_{ik,p} = 1 \quad \forall i \in V_1; \quad \forall p \tag{5}$$

$$\sum_{\substack{i=1\\ i\in V_2}}^{n}\sum_{k=1}^{r(i)} x_{ik}c_{ik} + \sum_{p=1}^{Maxnsub_i}\sum_{\substack{i=1\\ i\in V_1}}^{n}\sum_{k=1}^{r(i)} x_{ik,p}c_{ik,p} \leqslant C \tag{6}$$

$$\sum_{\substack{i=1\\ i\in V_2}}^{n}\sum_{k=1}^{r(i)} x_{ik}q_{ik} + \sum_{p=1}^{Maxnsub_i}\sum_{\substack{i=1\\ i\in V_1}}^{n}\sum_{k=1}^{r(i)} x_{ik,p}q_{ik,p} \geq Q \tag{7}$$

$$S_{n+1} \leqslant T \tag{8}$$

$$F_i + L_{ij} \leqslant S_j \quad \forall i,j \in V_2; \quad \forall (i,j) \in E_{FS} \tag{9}$$

$$F_{i,U_i} + L_{ij} \leqslant S_j \quad \forall i \in V_1; \quad \forall j \in V_2; \quad \forall (i,j) \in E_{FS} \tag{10}$$

$$F_i + L_{ij} \leqslant S_{j,1} \quad \forall i \in V_2; \quad \forall j \in V_1; \quad \forall (i,j) \in E_{FS} \tag{11}$$

$$F_{i,U_i} + L_{ij} \leqslant S_{j,1} \quad \forall i, \quad j \in V_1; \quad \forall (i,j) \in E_{FS} \tag{12}$$

$$S_i + L_{ij} \leqslant F_j \quad \forall i,j \in V_2; \quad \forall (i,j) \in E_{SF} \tag{13}$$

$$S_{i,1} + L_{ij} \leqslant F_j \quad \forall i \in V_1; \quad \forall j \in V_2; \quad \forall (i,j) \in E_{SF} \tag{14}$$

$$S_i + L_{ij} \leqslant F_{j,U_j} \quad \forall i \in V_2; \quad \forall j \in V_1; \quad \forall (i,j) \in E_{SF} \tag{15}$$

$$S_{i,1} + L_{ij} \leqslant F_{j,U_j} \quad \forall i,j \in V_1; \quad \forall (i,j) \in E_{SF} \tag{16}$$

$$S_i + L_{ij} \leqslant S_j \quad \forall i,j \in V_2; \quad \forall (i,j) \in E_{SS} \tag{17}$$

$$S_{i,1} + L_{ij} \leqslant S_j \quad \forall i \in V_1; \quad \forall j \in V_2; \quad \forall (i,j) \in E_{SS} \tag{18}$$

$$S_i + L_{ij} \leqslant S_{j,1} \quad \forall i \in V_2; \quad \forall j \in V_1; \quad \forall (i,j) \in E_{SS} \tag{19}$$

$$S_{i,1} + L_{ij} \leqslant S_{j,1} \quad \forall i,j \in V_1; \quad \forall (i,j) \in E_{SS} \tag{20}$$

$$F_i + L_{ij} \leqslant F_j \quad \forall i,j \in V_2; \quad \forall (i,j) \in E_{FF} \tag{21}$$

$$F_{i,U_i} + L_{ij} \leqslant F_j \quad \forall i \in V_1; \quad \forall j \in V_2; \quad \forall (i,j) \in E_{FF} \tag{22}$$

$$F_i + L_{ij} \leqslant F_{j,U_j} \quad \forall i \in V_2; \quad \forall j \in V_1; \quad \forall (i,j) \in E_{FF} \tag{23}$$

$$F_{i,U_i} + L_{ij} \leqslant F_{j,U_j} \quad \forall i,j \in V_1; \quad \forall (i,j) \in E_{FF} \tag{24}$$

$$\varepsilon_{ik} \leqslant t_{ik,p} \leqslant d_{ik} \quad \forall i \in V_1; \quad p = 1,2,\ldots,U_i+1; \quad k = 1,2,\ldots,r(i) \tag{25}$$

$$F_{i,p} - S_{i,p} = \sum_{k=1}^{r(i)} x_{ik,p}t_{ik,p} \quad \forall i \in V_1; \quad p = 1,2,\ldots,U_i+1 \tag{26}$$

$$F_i - S_i = \sum_{k=1}^{r(i)} x_{ik}d_{ik} \quad \forall i \in V_2 \tag{27}$$

$$S_{i,p+1} - F_{i,p} \leqslant \alpha_i \quad \forall i \in V_1; \quad p = 1,2,\ldots,U_i+1 \tag{28}$$

$$x_{ik,p+1} \leqslant x_{ik,p} \quad \forall i \in V_1 \tag{29}$$

$$\sum_{p=1}^{Maxnsub_i}\sum_{k=1}^{r(i)} \left(\frac{t_{ik,p}}{d_{ik}}\right) \times x_{ik,p} = 1 \quad \forall i \in V_1 \tag{30}$$

$$F_{i,p}, S_{i,p} \geqslant 0 \quad \forall i \in V_1; \quad p = 1,2,\ldots,U_i+1 \tag{31}$$

$$F_i, S_i \geqslant 0 \quad \forall i \in V_2 \tag{32}$$

$$F_0, S_0 = 0 \tag{33}$$

$$x_{ik}, x_{ik,p} \in \{0,1\} \quad \forall i,k,p \tag{34}$$

The relations (1)-(3) in the above formulation are used to describe time, cost, and quality objectives, respectively. Constraint (4) guarantees the selection of one and only one mode for each part of a non-preemptive activity. Constraint (5) guarantees the selection of one and only one mode for each preemptive activity. Constraint (6) defines the allowed upper-bound for the cost objective. Constraint (7) defines the allowed lower-bound for the quality objective. Constraint (8) defines the allowed upper-bound for the time objective. Constraints (9)–(12) preserve the precedence relations between the preemptive and non-preemptive activities with the finish-to-start relations. Constraints (13)–(16) preserve the precedence relations between the preemptive and non-preemptive activities with the start-to-finish relations. Constraints (17)–(20) preserve the precedence relations between the preemptive and non-preemptive activities with the start-to-start relation. Constraints (21)–(24) preserve the precedence relations between the preemptive and non-preemptive activities with the finish-to-finish relations. Constraint (25) guarantees that there is a minimum execution time ε_{ik} and a maximum execution time d_{ik} during which the

preemptive activity i in mode k is in progress without interruption. Constraint (26) shows that the duration of part p of preemptive activity i in mode k should be equal to the difference between the finish time and the start time for part p of activity i. Constraint (27) shows that the duration of non-preemptive activity i in mode k should be equal to the difference between the finish time and the start time of activity i. Constraint (28) guarantees that the interruption time for a given preemptive activity is less than or equal to a predefined level called α_i. Constraint (29), which is written for each part of a given preemptive activity in each mode, shows that, in a feasible solution, the later part of the activity will be scheduled if and only if the former part of the activity was scheduled. Constraint (30) ensures that the sum of the relative progress of all parts of a given preemptive activity in all execution modes is equal to unit. Constraints (31)–(34) guarantee that the decision variables are binary and positive.

3. Solution procedures

An efficient version of the ε-constraint method and a dynamic self-adaptive multi-objective evolutionary algorithm are proposed to generate different sets of non-dominated solutions for Model (1)–(34).

3.1. Efficient ε-constraint method for PGP-DTCQT problem

One of the advantages of the ε-constraint method is its ability to achieve efficient points in a non-convex Pareto curve (Chankong & Haimes, 1983). An efficient version of the ε-constraint method was proposed by Mavrotas, 2009. Khalili-Damghani, Tavana, and Sadi-Nezhad (2012) customized the augmented ε-constraint method proposed by Mavrotas (2009) for solving real-life large-scale multi-objective problems. Khalili-Damghani and Amiri (2012) proposed a multi-objective procedure based on the ε-constraint method to solve the multi-objective redundancy allocation problems. Khalili-Damghani, Abtahi, and Tavana (2013) also used an efficient version of the ε-constraint method to assess the performance of a multi-objective particle swarm optimization problem. The method proposed by Mavrotas (2009) is customized to generate the Pareto front for the PGP-DTCQT problem. The Model (35)–(38), presents an alternative formulation for Model (1)–(34):

$$\text{Min} \quad S_{n+1,0} + \beta \times (S_2/r_2 + S_3/r_3) \tag{35}$$

s.t.

$$\sum_{\substack{i=1\\ i\in V_2}}^{n}\sum_{k=1}^{r(i)} x_{ik}c_{ik} + \sum_{\substack{i=1\\ i\in V_1}}^{n}\sum_{k=1}^{r(i)} x_{ik,p}c_{ik,p} + S_2 = \varepsilon_2, \varepsilon_2 \in [Cost^-, Cost^+] \tag{36}$$

$$\sum_{\substack{i=1\\ i\in V_2}}^{n}\sum_{k=1}^{r(i)} x_{ik}q_{ik} + \sum_{p=1}^{Maxnsub_i}\sum_{\substack{i=1\\ i\in V_1}}^{n}\sum_{k=1}^{r(i)}(x_{ik,p}q_{ik,p}) - S_3 = \varepsilon_3, \varepsilon_3 \in [Quality^-, Quality^+] \tag{37}$$

$$X \in S \tag{38}$$

where r_2 and r_3 represent the range of the cost and quality objectives, respectively. The range of each objective is calculated by using the ideal and nadir values of the cost and quality objectives in the lexicographic payoff table of the original PGP-DTCQT model. $Cost^+$ and $Cost^-$ are, respectively, the ideal and nadir values for the cost objective of the single optimization problem of (2) and (4)–(36). Similarly, $Quality^+$ and $Quality^-$ are the ideal and nadir values of

Table 3
The parameters, notations, and decision variables.

Sets		
V	Set of activities	
V_1	Set of preemptive activities	
V_2	Set of non-preemptive activities	
E	Set of arcs	
Indices		
i, j	Index of activities	
k	Index of execution modes	
p	Index of parts activity parts	
Parameters		
n	Number of real project activities	
0, $n+1$	Dummy starting and ending activities	
C	Upper-bound of project cost	
T	Upper-bound of project time (deadline)	
Q	Lower-bound of project quality	
$Maxnpre_i$ (U_i)	Maximum number of allowed preemptions (maximum equals to $d_i - 1$)	$i \in V$
$Maxnsub_i$	Maximum number of activity parts (sub-activities) for an activity ($U_i + 1$)	$i \in V$
ε_{ik}	Minimum execution time of activity i without interruption in mode k	$i \in V$: $k = 1,2,\ldots,r(i)$
α_i	Maximum interruption time for activity i	$i \in V$
$r_{(i)}$	Number of execution modes for activity i	$i \in V$
c_{ik}	Cost of activity i in mode k	$i \in V_2$; $k = 1,2,\ldots,r(i)$
$c_{ik,p}$	Cost of Part p of activity i in mode k	$i \in V_1$; $k = 1,2,\ldots,r(i)$; $p = 1,2,\ldots,U_i$
q_{ik}	Quality of activity i in mode k	$i \in V_2$; $k = 1,2,\ldots,r(i)$
$q_{ik,p}$	Quality of Part p of activity i in mode k	$i \in V_1$; $k = 1,2,\ldots,r(i)$; $p = 1,2,\ldots,U_i$
L_{ij}	Time lag between activity i and activity j	$i \in V_1$; $j = 1,2,\ldots,n$
d_{ik}	Duration of activity i in mode k	$i \in V_2$; $k = 1,2,\ldots,r(i)$
Decision variables		
S_i	Positive decision variable: Start time for activity i	$i \in V_2$
F_i	Positive decision variable: Finish time for activity i	$i \in V_2$
$S_{i,p}$	Positive decision variable: Start time for part p of activity i	$i \in V_1$; $p = 1,2,\ldots,U_i$
$F_{i,p}$	Positive decision variable: Finish time for part p of activity i	$i \in V_1$; $p = 1,2,\ldots,U_i$
x_{ik}	Binary decision variable: Equals to 1 if activity i is executed in mode k; and 0 otherwise	$i \in V_2$; $k = 1,2,\ldots,r(i)$
$x_{ik,p}$	Binary decision variable: Equals to 1 if part p of activity i is executed in mode k; and 0 otherwise	$i \in V_1$; $k = 1,2,\ldots,r(i)$; $p = 1,2,\ldots,U_i$
$t_{ik,p}$	Positive continuous decision variable: Duration of part p of activity i in mode k	$i \in V_1$; $k = 1,2,\ldots,r(i)$; $p = 1,2,\ldots,U_i$

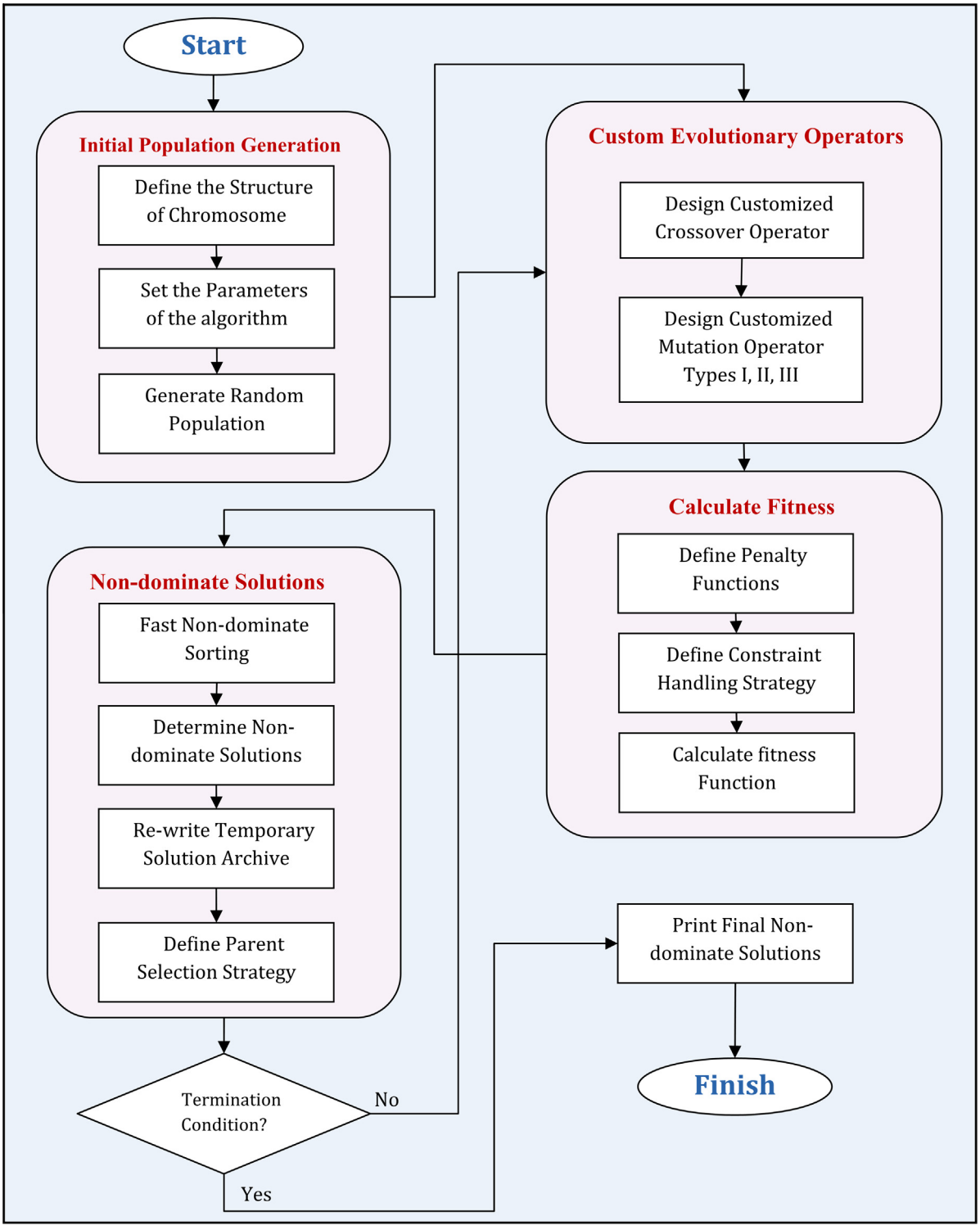


Fig. 1. The customized multi-objective evolutionary algorithm flowchart.

the quality objective function for the single optimization problem (3)–(34); where, $X \in S$ is the feasible region of the original PGP-DTCQT model (i.e., constraints (4)–(34)) and β is a positive value. It is notable that the term $(S_2/r_2 + S_3/r_3)$ results in the generation of strong non-dominated solutions on the Pareto front of the PGP-DTCQT model because formulations (35)–(38) produce solutions where the S_2 and S_3 values are near zero.

3.2. Customized multi-objective evolutionary algorithm

Non-dominated sorting genetic algorithm (NSGA) is an evolutionary algorithm used to solve the DTCTPs by producing a set of non-dominated solutions, Pareto, as the optimal solutions (Fallah-Mehdipour, Bozorg Haddad, Tabari, & Mariño, 2012). The classic NSGA-II algorithm proposed by Deb, Pratap, Agarwal, and

Number of interruptions				Number of interruptions				...	Number of interruptions			
Mode k	*Mode k*	...	*Mode k*	*Mode k*	*Mode k*	...	*Mode k*	...	*Mode k*	*Mode k*	...	*Mode k*
t_{1k1}	t_{1k2}	...	$t_{1k,ui}$	t_{2k1}	t_{2k2}	...	$t_{2k,ui}$	...	t_{ik1}	t_{ik2}	...	$t_{ik,ui}$

Fig. 2. The structure of a chromosome in the population.

```
Init_Pop ←Generate  P Random initial Solutions;
Current_Pop←Init_Pop
For i=1 to  Max_Itr_Num do
    Current_Pop ←Cross P_c% of current_Pop;
    Current_Pop ←Mutate P_m1% of Current_Pop using Type I ;
    Current_Pop ←Mutate P_m2% of Current_Pop using Type II ;
    Current_Pop ←Mutate P_m3% of Current_Pop using Type III ;
    Update_Parameters;
    Constraint_Handling;
    Fitness_Calc;
    Reposit_Archive← Current_Archive+ Current_Pop
    Fast_Pareto_Sort;
    Crowd_Dist;
    Current_Archive←Select Non-dominated solutions from Reposit_Archive
End for
```

Fig. 3. The high-level procedure map for the proposed MOEA.

0	1		2			0	3			
1	1	2	2	3	1	2	1	1	2	3
3	4	2	1	3	4	6	2	3	2	5

4a. A feasible solution (Parent I)

2			3				0	1		0
2	3	1	3	1	2	3	2	3	1	1
2	3	1	4	5	2	5	2	6	3	1

4b. A feasible solution (Parent II)

0	1		2			1	0
1	1	2	3	1	3	3	1	1
3	4	2	6	3	6	6	3	1

4c. Child I

2			3				0	0	3			
2	3	1	3	1	2	3	2	2	1	1	2	3
2	3	1	4	5	2	5	2	6	2	3	2	5

4d. Child II

Fig. 4. Cross-over operator.

0	1		2			0	3			
1	1	2	2	3	1	2	1	1	2	3
3	4	2	1	3	4	6	2	3	2	5

Parent

0	1		1		0	3			
1	1	2	2	3	2	1	1	2	3
3	4	2	1	3	6	2	3	2	5

Mutated solution child

5a. Mutation type I (Reduction)

0	1		2			0	3			
1	1	2	2	3	1	2	1	1	2	3
3	4	2	1	3	4	6	2	3	2	5

Parent

0	1		3				0	3				
1	1	2	2	3	1	2	2	1	1	2	3
3	4	2	1	3	4	1	6	2	3	2	5

Mutated solution child

5b. Mutation type I (Increment)

Fig. 5. Mutation operator (type I).

Meyarivan (2002) has been improved, customized, and extended to solve the proposed PGP-DTCQT model. Fig. 1 presents a schematic view of the customized NSGA-II algorithm proposed in this study.

A unique chromosome structure is proposed for a complete schedule of a project. The structure of the proposed chromosome can handle both preemptive and non-preemptive activities, concurrently. Considering a project with $n(i = 1, 2, \ldots, n)$ activities, $r(i)(k = 1, 2, \ldots, r(i))$, different execution modes for each activity, and a maximum of $p(p = 1, 2, \ldots, U_i)$ interruption possibilities, a

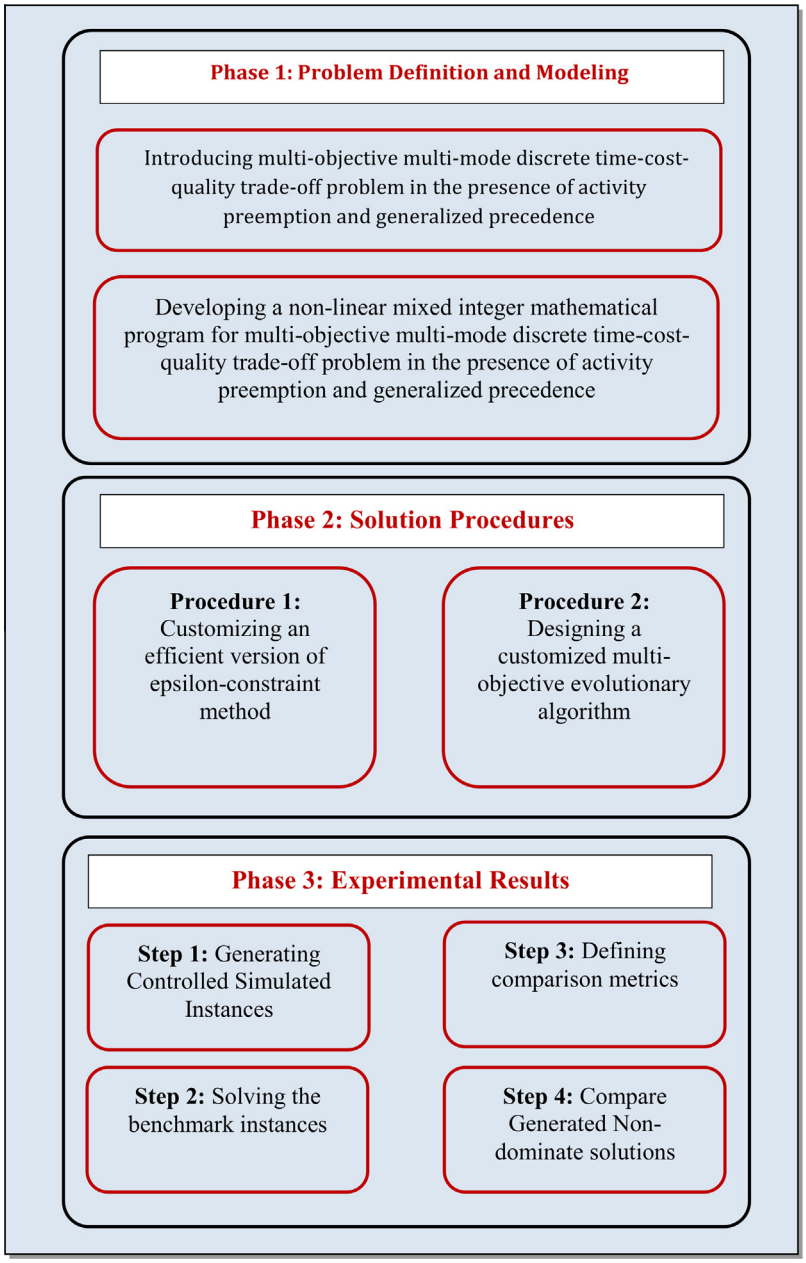


Fig. 6. The study process map.

triple line real coded chromosome is proposed. In the first line, an integer structure controls whether the activity is preemptive or non-preemptive. If the value is equal to zero the activity is non-preemptive. Hence, only one cell is reserved for the associated non-preemptive activity in lines 2 and 3. The positive value for line 1 represents a preemptive activity. Hence, a maximum number of $U_i + 1$ cells are reserved for the associated preemptive activity in lines 2 and 3. In the second line, the selected mode of part p of activity i is represented. For non-preemptive activities only part 1 will have positive value. The duration of part p for mode k of activity i is represented in line 3. Fig. 1 presents the general structure of a chromosome. As mentioned above, all alleles of the proposed structure in Fig. 2 may not be occupied in practice.

Fig. 3 presents the schematic view of the proposed algorithm for solving the instances of the PGP-DTCQT problem.

An activity can be interrupted in several different ways and the length of the chromosome is not constant because the number of preemptions is not a constant and it can vary in a range between 1 and a given maximum number of preemptions. Therefore, the genetic operators should be customized. Moreover the initial population should be constructed according to some basic properties.

Initial Population Generation. The possibility of preemption and the maximum number of preemptions for each activity are defined as parameters. The number of initial chromosomes is also used as a parameter. The number of execution modes for each activity as well as the processing time of each execution mode is parameters in the model. Line 1 of the chromosomes for non-preemptive activities is set to zero. The remaining cells in Line 1 of the chromosomes (i.e., preemptive activities) are valued randomly in the $[1, U_i]$ range using a discrete uniform distribution function, where U_i is the maximum allowed number of preemption for activity i. Then, line 2 of the chromosomes is filled randomly in the $[1, k]$ range using a discrete uniform distribution function, where k is the execution mode of the activity. As mentioned above, line 3

Algorithm 1: Problem Generation

For i=1 **to** *number of project network arcs* **do**
  Generate generalized precedence relations using Algorithm 2;
  Assign time lags to each arc using algorithm 3;
End for
For i=1 **to** number of project activities **do**
  Generate execution modes using algorithm 4;
End for

Algorithm 2: Generalized Precedence Relation and Time Lags Generation

For i=1 **to** *number of project network arcs* **do**
  $N_p \leftarrow$ number of relation types
  $P \leftarrow$ Random number in $U[1, N_p] \in Z^+$
 PT $\leftarrow$ Corresponding relation type with respect to N_p
 Arc(i) relation type $\leftarrow$ PT
End for

Algorithm 3: Time Lag Assigning

For i=1 **to** *number of project network arcs* **do**
  Let 1: negative time lag
  Let 2: positive time lag
  $R \leftarrow$ Random number in $U[1,2] \in Z^+$
  $S \leftarrow$ Corresponding time lag sign with respect to R
  $\text{TimeLag}_{Max} \leftarrow$ Maximum allowed time lag
  $TL \leftarrow$ Random number in $U[1, \text{TimeLag}_{Max}] \in Z^+$
  Arc(i) time lag $\leftarrow$ TL with respect to sign of S
End for

Algorithm 4: Execution Mode Generation

For i=1 **to** *number of project activities* **do**
  $\text{ModeNumber}_{Max} \leftarrow$ Maximum allowed number of mode for activity
  $\text{ActivityEM}(i) \leftarrow$ Random number in $U[2, \text{ModeNumber}_{Max}] \in Z^+$
  For j=1 **to** $\text{ActivityEM}(i)$ **do**
    $\text{ActivityTime}_{Min} \leftarrow$ Minimum allowed activity time
    $\text{ActivityTime}_{Max} \leftarrow$ Maximum allowed activity time
    $\text{ModeTime}(j) \leftarrow$ Random number in $U[\text{ActivityTime}_{Min}, \text{ActivityTime}_{Max}] \in Z^+$
  End for
  Sort activity(i) modes based on their time in ascending order
  $\text{ActivityCost}_{Min} \leftarrow$ Minimum allowed activity cost
  $\text{ActivityCost}_{Max} \leftarrow$ Maximum allowed activity cost
  $\text{ModeCost}(1) \leftarrow$ Random number in $U[\text{ActivityTime}_{Min}, \text{ActivityTime}_{Max}] \in Z^+$
  For j=2 **to** $\text{ActivityEM}(i)$ **do**
    $S \leftarrow$ Random number in $U[1, 5] \in Z^+$
    $\text{ModeCost}(j) \leftarrow \text{ModeCost}(j\text{-}1) - S\,(\text{ModeTime}(j) - \text{ModeTime}(j\text{-}1))$
  End for
  ModeQuality(1)=0.99
  For j=2 **to** $\text{ActivityEM}(i)$ **do**
    Let 1: 0.95…0.99
    Let 2: 0.90…0.99
    Let 3: 0.85…0.99
    Let 4: 0.80…0.99
    Let 5: 0.75…0.99
    QL $\leftarrow$ Random number in $U[1, 5] \in Z^+$
    $\text{ModeQuality}(j) \leftarrow$ Random number in range of QL
  End for
End for

Fig. 7. The generalized precedence relations and the execution mode generation algorithms.

holds the duration of part p of the *k-th* execution mode for activity i. Line 3 is filled randomly in the $[1, d_{ik}]$ range using a discrete uniform distribution function. It is notable that lines 2 and 3 only have one cell for non-preemptive activities. Line 2 of the non-preemptive activities is filled the same as preemptive activities. But the line 3 for non-preemptive activities is filled using the associated single value d_{ik}.

Complete Schedule. A complete schedule for proposed PGP-DTCQT model is a schedule in which the following three properties are illustrated for all activities: (a) The number of interruptions for each preemptive activity is determined; (b) The execution modes for all activities are determined; and (c) The processing times of all parts for all preemptive activities and the processing time of all non-preemptive activities are determined. It is notable that a complete schedule can be infeasible. The feasibility of a schedule is checked based on the constraint handling procedure proposed in Section 5.2.2.

Four dependent types of genetic operators are considered. All types have a tunable positive portability of implementation in order to improve the exploration and exploitation of the algorithm.

Crossover operator. A single point cut is used as crossover operator based on line 1 of the chromosome. Based upon a crossover probability, called P_C, two parents are selected and the crossover is accomplished. The cut point is randomly determined uniformly

Table 4
The characteristics of the networks of generated random instances.

Instance number	Number of activities	Number of arcs	Order strength*	Complexity index**
1	10	45	1	5
2	50	1225	1	42
3	100	4950	1	87

* The number of precedence relations divided by the theoretical maximum number of precedence relations in the network.
** The measure of the closeness of a network to a series-parallel directed graph.

Table 5
The fitted parameters of the solution procedures.

Efficient ε-constraint	Test instances		
	Small	Medium	Large
Archive size	30	30	30
Upper-bound of cost	31,397	153,579	314,310
Upper-bound of quality	0.984	0.9786	0.9774
Lower-bound of cost	29,590	143,016	293,269
Lower-bound of quality	0.834	0.8222	0.8327
Step Size of cost	0.05	0.05	0.05
Step Size of quality	0.04	0.04	0.04
NSGA-II	Test Instances		
	Small	Small	Small
Chromosome number	200	200	200
Archive size	30	30	30
Maximum iteration number	1000	1000	1000
Cross rate	[0.8,0.7]	[0.8,0.7]	[0.8,0.7]
Mutation rate ($p_{mi}, i = 1,2,3$)	[0.02,0.04]	[0.02,0.04]	[0.02,0.04]
Alpha	1	1	1
Beta	5	5	5

in the range of the number of activities. Fortunately, this type of crossover has no effect on the feasibility of the reproduced solutions.

Consider Fig. 4 for an explanation of the crossover operator. Assume that a project has five activities, with a maximum of three interruptions and three possible execution modes per activity, a minimum execution time of each part of activity without interruption which is assumed equal to 1, and a maximum execution time of each part of activity without interruption which is assumed to be equal to d_{ik}.

Solutions 4a and 4b present the schematic structure of the chromosome for two arbitrary solutions for this project. The represented solution for chromosome 4a has five activities. The number of interruptions in the activities is 0, 1, 2, 0, and 3, respectively. For example Activity 3 has two interruptions. This will divide Activity 3 into three parts. These three parts are implemented in mode 2, 3, and 1, respectively. The execution time of the parts is 1, 3, and 4, respectively. The other activities have similar descriptions. Solution 4b has a similar description. A single crossover has been accomplished on the chromosome in Fig. 4a and b. If the cut-point is assumed to be equal to 3, the resulting children are represented by Solutions 4c, and 4d.

Mutation operator (type I). The value of a randomly selected cell in line 1 of a given chromosome is changed uniformly in the interval of $[0, U_i]$ based on the mutation probability, called P_{m1}. Lines 2 and 3 are re-structured and modified based on the changes made in line 1. The modifications usually are accomplished randomly. Fig. 5 illustrates the mechanism for this operator.

More formally, the mutation operator type I has two distinct sub-types (i.e., reduction, and increment) as shown in Fig. 5a and b. It is notable that the possibility for preemption of an activity is naturally one of the substantial properties of the activity. Therefore, the mutation operator type I essentially is done for activities which have the possibility of the preemption. The illustrative examples in Fig. 4 are provided to demonstrate the mechanism of the operator. If the associated value of a cell in line 1 is set to zero, it means that the associate activity cannot be preempted and the mutation operator type I does not act on such cells. This type of mutation is strong enough to change the structure of a schedule. Hence, it can be used with a higher rate in the exploration phase.

Mutation operator (type II). The value of a randomly selected cell in line 2 of a given chromosome is changed uniformly in the range of execution modes based on the mutation probability defined as P_{m2}. Consequently, line 3 of the chromosome should be modified. A checking procedure is provided to check the feasibility of the associated cell in line 3 considering the processing time of the new execution mode. Since the number of infeasible solutions may increase extensively for medium and large size instances, the infeasible solutions are modified in this operator. If the total processing time of the new execution mode is less than the processing time of part p (i.e., mutated cell) of the old execution mode, the associated cell in line 3 is then randomly selected in range $[1, d_{ik}]$ based on a uniform distribution function where d_{ik} is the duration of Activity i in its new selected execution mode (i.e., after mutation). If the total processing time of the new execution mode is greater than the processing time of part p (i.e., mutated cell) of the old execution mode, the associated cell in line 3 is left unchanged.

Mutation operator (type III). Two cells in line 3 of a given chromosome are selected randomly based on the mutation probability P_{m3}. The values of the selected cells are replaced with each other.

P_c and P_{mi}; $i = 1, 2,$ and 3; are determined dynamically using linear equations which relate the crossover and mutation rate to the iteration number in the algorithm. P_c is set close to 1 in the first iteration of the algorithm and is decreased as the iterations proceed. P_{mi} is configured to a small value in the first iteration of the algorithm and is increased until the last iteration. The dynamic parameter tuning causes better exploration and exploitation. The fitness function is calculated based on a triple vector of the objective functions.

3.3. Constraint handling strategy for the customized NSGA-II method

The time, cost, and quality windows in model (1)–(35) are satisfied using a heuristic self-adaptive penalty approach. The generalized precedence constraints are handled through the elimination approach.

The value of the objective function may violate the time, cost, and quality window constraints. In these situations the value of the violation for each chromosome in the population is calculated with relations (39)–(41) as follows:

$$v_{i1} = \text{Max}\{0, Time - T\} \quad (39)$$

$$v_{i2} = \text{Max}\{0, Cost - C\} \quad (40)$$

$$v_{i3} = \text{Max}\{0, Q - Quality\} \quad (41)$$

where, v_{i1}, v_{i2}, and v_{i3} are the violation values associated with the time, cost, and quality of chromosome i, respectively. T and C are the upper-bound of the time and the cost objectives, respectively. Q is the lower-bound of the quality objective.

Based on the violation values, the minimum value of the violations, the iteration number of the algorithm, and the dynamic self-adaptive penalty values are calculated with Eqs. (42)-(44) as follows:

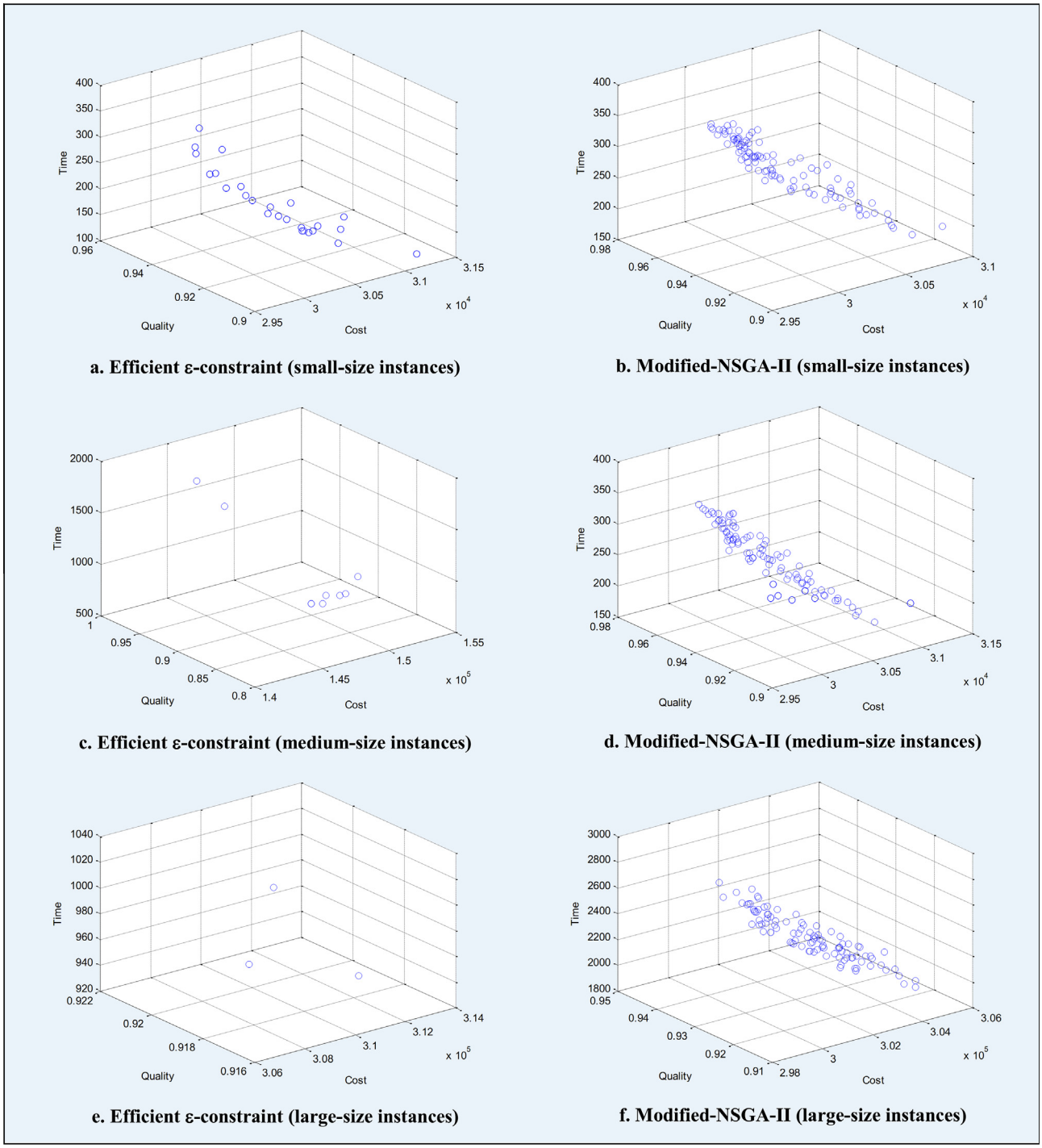


Fig. 8. The generated non-dominated solutions for both methods.

$$TimePenalty = \left(\frac{v_{i1}}{v_{i1}^{min}}\right)^{\alpha} \times t^{\beta} \quad (42)$$

$$CostPenalty = \left(\frac{v_{i2}}{v_{i2}^{min}}\right)^{\alpha} \times t^{\beta} \quad (43)$$

$$QualityPenalty = \left(\frac{v_{i3}}{v_{i3}^{min}}\right)^{\alpha} \times t^{\beta} \quad (44)$$

where, v_{i1}, v_{i2}, and v_{i3} have the same definition of (41)-(43). $v_{i1}^{\min} = \min_i\{\varepsilon + v_{i1}\}$, $v_{i2}^{\min} = \min_i\{\varepsilon + v_{i2}\}$, and $v_{i3}^{\min} = \min_i\{\varepsilon + v_{i3}\}$ are the minimum violation values for the time, cost, and quality of the chromosomes in the population, respectively. The constant ε is a small positive value to avoid division by zero, t is the iteration number, and α and β are the control parameters. The penalty values are applied to the associated fitness functions as (45)-(47), respectively.

$$Time_i' = Time_i + Time\ Penalty \quad (45)$$

$$Cost_i' = Cost_i + Cost\ Penalty \quad (46)$$

$$Quality_i' = Quality_i - Quality\ Penalty \quad (47)$$

Table 6
Comparison metrics (Coello et al., 2007; Yu and Gen, 2010).

Type	Metric	Description	Criterion
Convergence metrics	NNS	The number of non-dominated solutions found by each procedure	N/A
	ER	Error rate (used to measures the non-convergence of the procedure towards the real Pareto front)	$ER = \frac{\sum_{i=1}^{n} e_i}{n}$ $e_i = \begin{cases} 0 & S\text{olution i belongs to Pareto front} \\ 1 & \text{otherwise} \end{cases}$
	GD	Generational distance (used to calculates the distance between the reference set and the solution set generated by a procedure)	$GD = \frac{\sum_{i=1}^{n} d_i}{n}$ $d_i = \underset{p \in PF}{Min} \left\{ \sqrt{\sum_{k=1}^{m} \left(Z_k^i - Z_k^i \right)^2} \right\}$ where m is the number of objective values
Distribution metrics	SM	Spacing metric (used to measure the uniformity of the spread of the points of the solution set generated by a procedure)	$\bar{d} = \frac{\sum_{i=1}^{n} d_i}{n}$ $SP = \sqrt{\frac{\sum_{i=1}^{n} (\bar{d} - d_i)^2}{n-1}}$
	DM	Diversification metric (used to measure the spread of the solution set generated by a procedure)	$DM = \left[\sum_{i=1}^{n} Max(x_i - y_i) \right]^{\frac{1}{2}}$ where $\|x_i - y_i\|$ is the Euclidean distance between two non-dominated solutions x_i and y_i

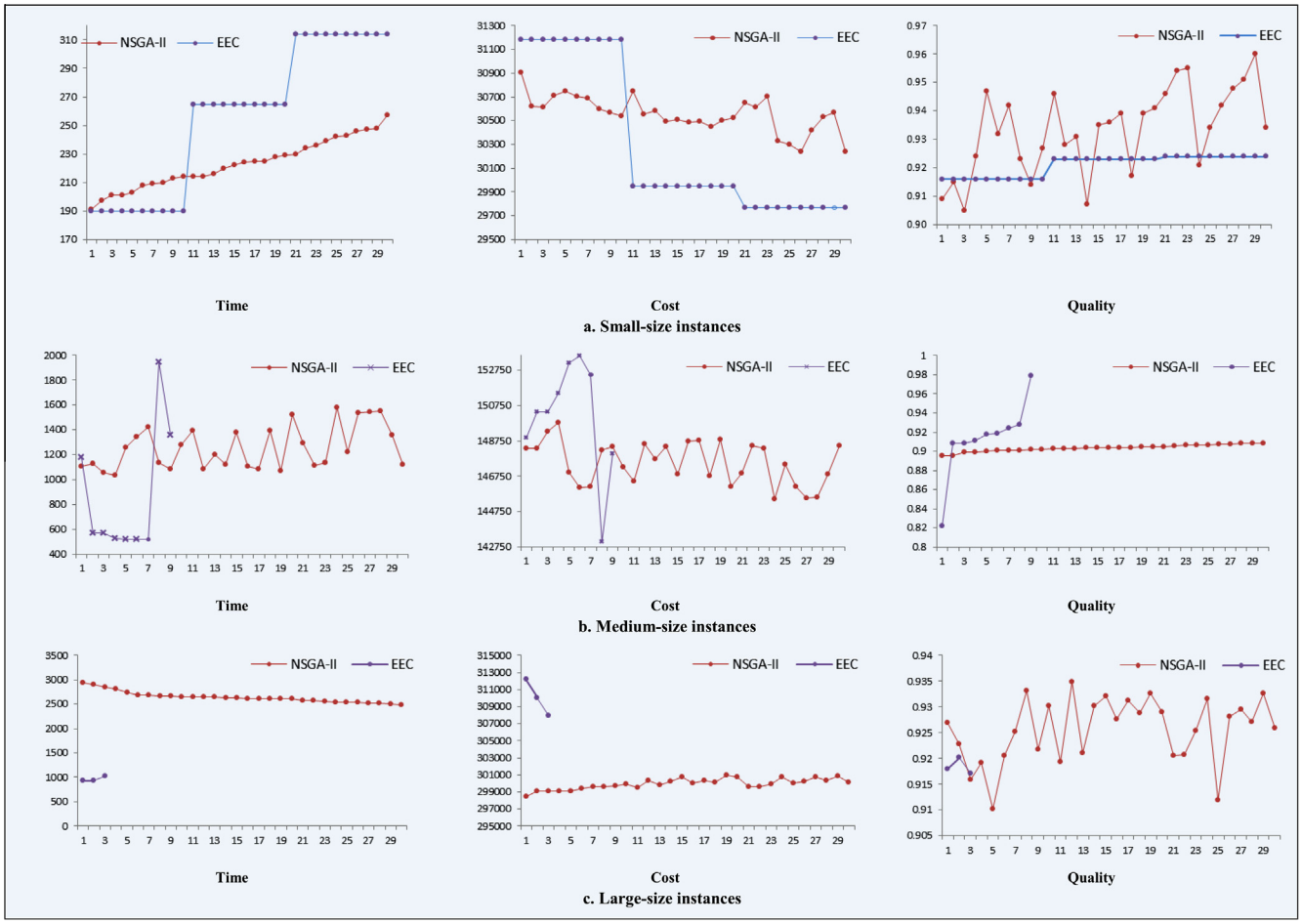


Fig. 9. The time, cost, and quality objectives.

where $Time_i'$, $Cost_i'$, and $Quality_i'$ are the penalized time, cost, and quality objectives for the violated chromosome i. Those chromosomes which do not satisfy the generalized precedence constraints are eliminated during the population initialization and evolutionary operators. Fig. 6 presents a schematic view of the process map in this study.

4. Computational experiments

4.1. Problem generation

We utilized simulation to investigate the performance of the two solution procedures described in the previous section. A

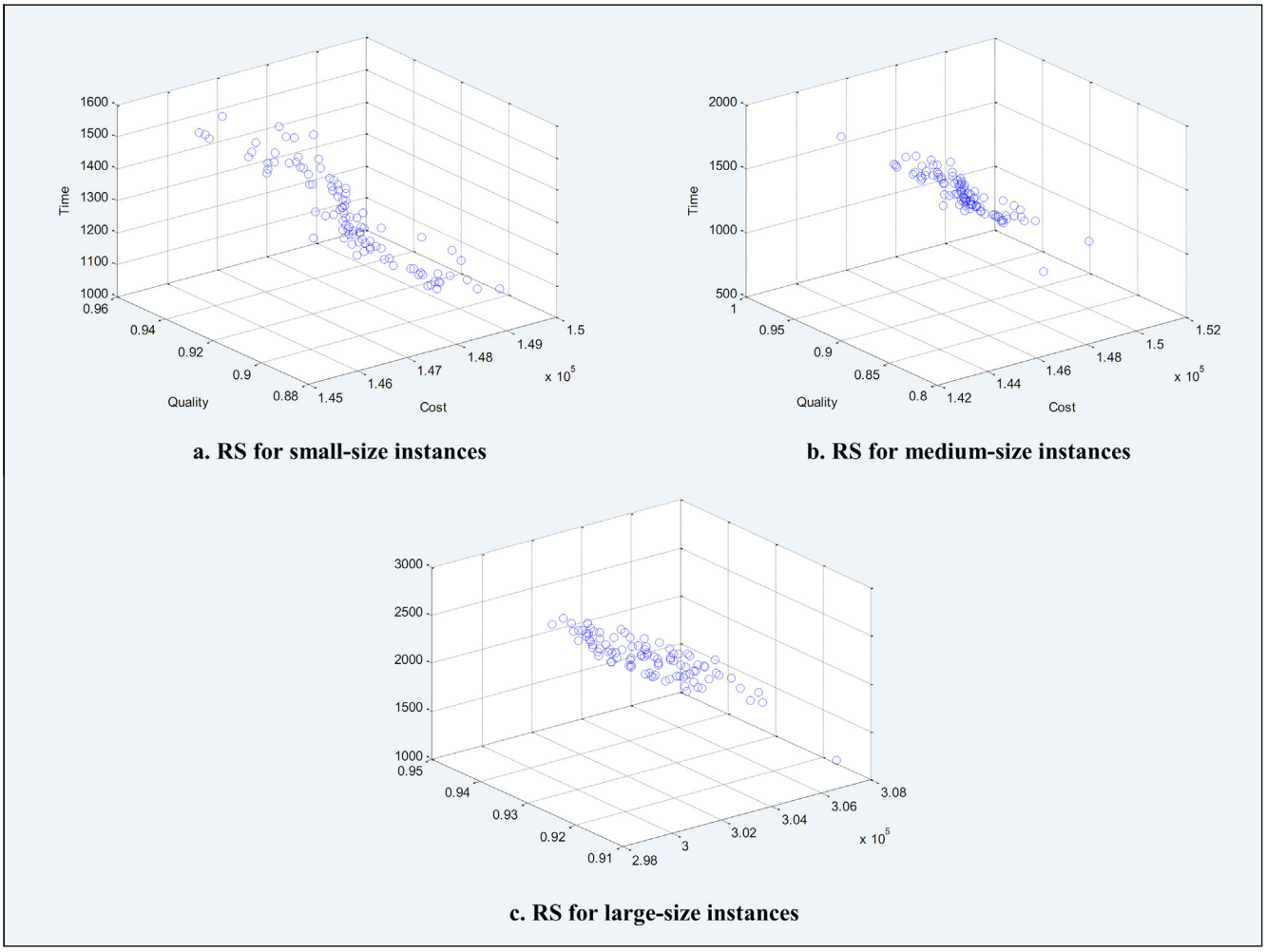


Fig. 10. The reference sets (RSs) for three different benchmark instances.

three-phase procedure was used in the simulation model to generate the network design, the generalized precedence relations, and the activities' execution modes. The RaGEN software proposed by Demeulemeester, Vanhoucke, and Herroelen (2003) was utilized to generate the networks of instances characterized in Table 4.

In addition to the parameters characterized in Table 4, we set $U_i = 3$, $\varepsilon_{ik} = 2$, and $\alpha_i = 2$. We extended the procedure proposed by Tareghian and Taheri (2007) to assign generalized precedence relations, time lags, and execution modes to each activity in the projects. The Pseudo-code of the proposed procedure is illustrated for Algorithm 1 to Algorithm 4 in Fig. 7.

Three classes of instances (i.e., small size, medium size, and large size) were generated in order to check and compare the performance of the proposed solution procedures. We have generated 10 benchmark instances in each class. Each instance has been solved 10 times by each of the proposed solution procedures. The mean values of all runs and all instances in a class has been plotted and discussed. Therefore, the quality, time, cost, and CPU times are the mean values of all runs for all instances for different class of problems.

4.2. Software–hardware implementation and parameter tuning

The proposed ε-constraint method was coded using LINGO 13.0. The dynamic self-adaptive multi-objective evolutionary algorithm was coded using Visual Basic 6.0. All codes were run on a Pentium IV PC with MS-Windows 7, 4 GB of RAM, and 2.2 GHz Core 2 Due CPU. The parameters were tuned for both methods based on experimental analysis. The fitted values for the parameters are illustrated in Table 5.

4.3. Comparison metrics

Two sets of accuracy and diversity metrics were utilized to provide a basis for evaluating the relative performance of both solution procedures. These metrics, presented in Table 6, have been proposed for an extensive comparison of multi-objective optimization procedures by Coello, Lamont, and Veldhuizen (2007) and Yu and Gen (2010).

4.4. Results

The efficient ε-constraint method and dynamic self-adaptive multi-objective evolutionary algorithm were implemented on all generated instances. Fig. 8 represents the generated non-dominate solutions for both methods for all instances.

A careful examination of Fig. 8 reveals that in all benchmark instances, the generated Pareto fronts of the efficient ε-constraint method are sparse while these are dense in the dynamic self-adaptive multi-objective evolutionary algorithm. The situation is more critical for large-size instances as the ε-constraint method

Fig. 11. The achieved objective functions for each method vs. the RS for small size instances.

generates a limited number of distinct non-dominate solutions over the real Pareto front.

The CPU time of the ε-constraint method was 300, 2400, and 3000 s for small, medium, and large size instances, respectively. The CPU time of the dynamic self-adaptive NSGA-II method was 40, 130, and 400 s for small, medium and large size instances, respectively.

Fig. 9 presents the comparative analysis of different objective functions for thirty non-dominated solution samples on both methods for all categories of the benchmark instances. As shown

Table 7
The computational results for the accuracy and diversity metrics.

Run	Accuracy metrics						Diversity metrics				Run time	
	NNS		ER		GD		SM		DM		CPU time (S)	
	NSGA II	ε Constraint	NSGA II	ε Constraint	NSGA II	ε Constraint	NSGA II	ε Constraint	NSGA II	ε Constraint	NSGA II	ε Constraint
Small-size instances												
1	74	30	0.26	0.7	4.24	2313.93	17.03	8332.97	271.324	2060.743	13	121.524
2	54	40	0.46	0.6	13.34	98.66	59.82	260.50	265.4091	2060.743	14.6	119.767
3	37	50	0.63	0.5	10.95	17.85	25.63	45.88	292.7654	2060.743	13.688	120.053
4	58	50	0.42	0.5	6.55	20.91	20.53	51.82	297.739	2060.743	13.313	120.592
5	74	50	0.26	0.5	3.11	28.59	12.92	74.13	302.0842	2060.743	13.797	120.468
6	78	50	0.22	0.5	2.17	22.15	8.61	55.37	309.103	2060.743	13.617	121.298
7	71	50	0.29	0.5	4.50	25.04	22.31	65.92	314.0928	2060.743	13.484	120.622
8	70	50	0.3	0.5	4.96	24.96	18.90	65.87	314.7681	2060.743	13.016	120.647
9	61	50	0.39	0.5	5.97	12.43	21.19	31.66	316.334	2060.743	13.359	121.263
10	69	50	0.31	0.5	6.52	13.58	23.76	35.27	318.0094	2060.743	12.828	120.279
Ave.	**64.6**	**47**	**0.354**	**0.53**	**6.231**	**257.81**	**23.07**	**901.939**	**300.162**	**2060.743**	**13.4702**	**120.6513**
Std. Dev.	**12.3666**	**6.749486**	**0.12366**	**0.067495**	**3.46199**	**722.8799**	**13.881**	**2611.845**	**18.7707**	**4.79E-13**	**0.508736**	**0.564757**
Medium-size instances												
1	90	21	0.1	0.79	219.84	7304.30	829.30	8863.22	507.8854	2331.905	77.45768	3600
2	63	21	0.37	0.79	148.23	7304.30	592.92	8863.22	455.4259	2331.905	57.75	3600
3	78	21	0.22	0.79	116.42	7304.30	399.11	8863.22	521.9694	2331.905	74.15625	3600
4	67	21	0.33	0.79	66.98	7304.30	276.75	8863.22	617.5348	2331.905	54.67188	3600
5	38	20	0.62	0.83	83.99	7304.30	265.93	8863.22	632.0249	2331.905	56.25	3600
6	34	20	0.66	0.83	47.57	7304.30	135.49	8863.22	636.15	2331.905	56.77344	3600
7	13	21	0.87	0.79	282.90	7304.30	657.60	8863.22	636.9724	2331.905	56.64844	3600
8	1	21	0.99	0.79	2141.57	7304.30	5233.09	8863.22	3448.319	2331.905	54.9375	3600
9	33	21	0.67	0.79	108.67	7304.30	290.69	8863.22	3448.366	2331.905	56.85156	3600
10	42	21	0.58	0.79	69.15	7304.30	199.30	8863.22	3448.631	2331.905	57.42188	3600
Ave.	**45.9**	**20.8**	**0.541**	**0.798**	**328.529**	**7304.303**	**888.017**	**8863.225**	**1435.32**	**2331.905**	**60.29186**	**3600**
Std. Dev.	**28.2781**	**0.421637**	**0.28278**	**0.016865**	**641.289**	**1.82E-12**	**1542.73**	**1.92E-12**	**1390.54**	**0**	**8.270859**	**0**
Large-size instances												
1	99	11	0.01	0.89	6.277778	8069.766	108.97	10497.25	585.0552	2943.12	214.9922	7200
2	53	20	0.47	0.8	42.32	8000.00	120.77	10403.62	672.734	2943.123	213.4023	7200
3	51	1	0.49	0.99	108.66	9340.44	543.70	13685.74	705.9236	2943.123	215.0703	7200
4	55	4	0.45	0.96	40.25	8161.70	113.34	10605.17	713.9622	2943.123	212.8164	7200
5	97	13	0.03	0.87	4.097599	8039.869	42.65	10451.96	746.8129	2943.12	217.0586	7200
6	82	20	0.18	0.8	12.6312	8000	60.72	10403.62	747.994	2943.12	210.2031	7200
7	68	20	0.32	0.8	42.39	8000.00	155.05	10403.62	758.5173	2943.123	210.9453	7200
8	88	4	0.12	0.96	8.586149	8161.698	44.80	10605.17	772.8406	2943.12	214.7539	7200
9	17	4	0.83	0.96	182.68	8161.70	1633.90	10605.17	5474.265	2943.123	219.2031	7200
10	47	4	0.53	0.96	55.56	8161.70	149.68	10605.17	5474.265	2943.123	218.1641	7200
Ave.	**65.7**	**10.1**	**0.343**	**0.899**	**50.3465**	**8209.687**	**297.358**	**10826.65**	**1665.23**	**2943.123**	**214.6609**	**7200**
Std. Dev.	**25.9702**	**7.709302**	**0.25970**	**0.077093**	**56.1914**	**403.8901**	**491.514**	**1008.65**	**2008.25**	**4.79E-13**	**2.936407**	**0**

in Fig. 9, the number of generated distinct non-dominate solutions for the ε-constraint method is very low for medium and large scale instances.

A reference set (RS) was defined for each benchmark instance since the real Pareto fronts of the benchmark instances were not known in advance. A RS is the set of best known solutions of a benchmark instance throughout the 10 different runs for both methods. Fig. 10 presents the RSs for three categories of benchmark instances.

As the number of generated solutions by the ε-constraint method is not meaningful in the medium and large scale instances, Fig. 11 only presents the achieved objective functions for each method against the RS for small size instances. This may help to recognize the relative effect of each method in the construction of RS. Plotting the achieved solutions against the RS in medium and large size instances has no practical use since the RSs are approximately the same as the solutions generated by the dynamic self-adaptive NSGA-II method.

The comparison metrics have been calculated for both methods using the RSs for all benchmark instances over 10 different runs. Table 7 presents the computational results of the accuracy and diversity metrics for small, medium, and large size instances.

As shown in Table 7, the dynamic self-adaptive NSGA-II method outperforms the ε-constraint method with regards to all comparison metrics. The dynamic self-adaptive NSGA-II method has a better CPU time for all runs and all benchmark instances. The Number of Non-dominated Solutions (NNS) generated by the NSGA-II method is greater than the NNS generated by the ε-constraint method for all runs and all instances. Since the NSGA-II method has a lower error rate (ER) in comparison with the ε-constraint method in all runs, it has a relatively lower non-convergence towards the RS. The proposed NSGA-II method has less Generated Distance (GD) in all runs. This shows that the Non-dominated Solutions generated by the proposed NSGA-II method are closer to the RS in comparison with the ε-constraint method. The value of the NNS, ER, and GD metrics on different instances reveals that the NSGA-II method outperforms the ε-constraint method in accuracy.

The NSGA-II method also outperformed the ε-constraint method in the diversification metrics. As shown in Table 7, the SM and DM for the NSGA-II method are greater than for the ε-constraint method. The greater Spacing Metric (SM) revealed that the NSGA-II method generated more uniformly distributed solutions throughout the RS in comparison with the ε-constraint method. In addition, larger DMs indicated that the solution set generated by the NSGA-II method covers more space of the RS in comparison with the ε-constraint method. Moreover, as shown in Table 7, the CPU times of the ε-constraint method are considerably longer than those of the NSGA-II method in all instances. The ε-constraint method was inapplicable specifically in medium and large scale problems.

5. Conclusions and future research directions

Trade-off problems try to schedule activities of the project in a way to obtain a nice balance between cost, time, and quality measurements. Finding the non-dominated solutions considering time, cost and quality objectives can be interesting in trade-off problems. In the real world, there are several generalized types of precedence relations between the activities of a project. Moreover, an activity may be interrupted in one time period and re-started in another time period of the project in a different mode. It is clear that a maximum number of interruptions are acceptable in practical environments. An interrupted activity should also be re-started no later than a pre-defined interval of time. Otherwise, it is assumed as a re-work and should be re-scheduled. Moreover, an activity should run a minimum number of times without preemption in each planning horizon. Considering these and the fact that in practice the resources are used in discrete units in projects, a new multi-objective multi-mode project time–cost–quality trade-off model under preemption and generalized precedence relations was developed.

Two multi-objective solution procedures were proposed to solve the PGP-DTCQT model. One is based on mathematical programming and is called the efficient ε-constraint which is a customized version of the classical ε-constraint method. The efficient ε-constraint method has several advantages over the classical ε-constraint method. The calculation of the range for the objective functions over the efficient set was accomplished with a lexicographic payoff table. The search of the solution space was done systematically to decrease the solution time for problems with more than two objective functions. The other one is based on a dynamic self-adaptive multi-objective evolutionary algorithm. A special chromosome structure was designed for the problem. Customized evolutionary operators were designed for PGP-DTCQT model. The proposed algorithm was also equipped with a self-adaptive penalty function and a dynamic parameter-tuning function. The dynamic self-adaptive penalty function penalized the violated chromosomes according to the status of the chromosome, the status of the entire population, and the iteration of the algorithm. Through dynamic parameter-tuning, the crossover, mutation and penalty parameters were modified based on the iteration number in favor of a better exploration and exploitation.

In order to provide a basis for comparison and investigating the efficiency and applicability of the two procedures, we systematically simulated a series of random problems. The generated Pareto fronts of both methods were compared against reference sets (i.e., the best known Pareto fronts of instances) based on two sets of diversification and accuracy metrics. The proposed dynamic self-adaptive multi-objective evolutionary algorithm represented relative dominance in comparison with the efficient ε-constraint method.

Acknowledgement

The authors would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions.

References

Afshar, A., Kaveh, A., & Shoghli, O. (2007). Multi-objective optimization of time–cost–quality using multi-colony ant algorithm. *Asian Journal of Civil Engineering (Building and Housing), 8*(2), 113–124.

Azaron, A., Perkgoz, C., & Sakawa, M. (2005). A genetic algorithm approach for the time–cost trade-off in PERT networks. *Applied Mathematics and Computation, 168*(2), 1317–1339.

Babu, A. J., & Suresh, N. (1996). Project management with time, cost and quality considerations. *European Journal of Operational Research, 88*, 320–327.

Ballestín, F., Valls, V., & Quintanilla, S. (2008). Preemption in resource-constrained project scheduling. *European Journal of Operational Research, 189*, 1136–1152.

Ballestín, F., Valls, V., & Quintanilla, S. (2009). Scheduling projects with limited number of preemptions. *Computers and Operations Research, 36*, 2913–2925.

Baptiste, P., & Demassey, S. (2004). Tight LP bounds for resource constrained project scheduling. *OR Spectrum, 26*(2), 251–262.

Bianco, L., & Caramia, M. (2012). An exact algorithm to minimize the makespan in project scheduling with scarce resources and generalized precedence relations. *European Journal of Operational Research, 219*, 73–85.

Błażewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Węglarz, J. (2007). *Handbook on scheduling: From theory to applications*. Berlin: Springer Verlag.

Burns, S., Liu, L., & Feng, C. (1996). The LP/IP hybrid method for construction time–cost trade-off analysis. *Construction Management and Economics, 14*(3), 265–276.

Chankong, V., & Haimes, Y. (1983). *Multi-objective decision making theory and methodology*. New York: Elsevier Science.

Chao-guang, J., Shang, J., Yan, L., Yuan-min, Z., & Zhen-dong, H. (2005). Research on the fully fuzzy time–cost trade-off based on genetic algorithms. *Journal of Marine Science and Application, 4*(3), 18–23.

Chen, R. M. (2011). Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. *Expert Systems with Applications, 38*(6), 7102–7111.

Chen, R. M., Wu, C. L., Wang, C. M., & Lo, S. T. (2010). Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. *Expert Systems with Applications, 37*(3), 1899–1910.

Coello, C. A. C., Lamont, G. B., & Veldhuizen, D. A. V. (2007). *Evolutionary algorithms for solving multi-objective problems*. New York: Springer.

Correia, I., Lourenço, L. L., & Saldanha-da-Gama, F. (2012). Project scheduling with flexible resources: Formulation and inequalities. *OR Spectrum, 34*(3), 635–663.

Crandall, K. C. (1973). Project planning with precedence lead/lag factors. *Project Management Quarterly, 4*(3), 18–27.

De, P., Dunne, E., Ghosh, J., & Wells, C. (1997). Complexity of the discrete time/cost trade-off problem for project networks. *Operations Research, 45*, 302–306.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast an elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Demeulemeester, E., De Reyck, B., & Herroelen, W. (2000). The discrete time/resource trade-off problem in project networks: A branch-and-bound approach. *IIE Transactions, 32*(11), 1059–1069.

Demeulemeester, E., & Herroelen, W. (1996). An efficient optimal procedure for the preemptive resource-constrained project scheduling problem. *European Journal of Operational Research, 90*, 334–348.

Demeulemeester, E., & Herroelen, W. (2002). *Project scheduling: A research handbook*. Dordrecht: Kluwer Academic Publishers.

Demeulemeester, E., Vanhoucke, M., & Herroelen, W. (2003). RanGen: A random network generator for activity-on-the-node networks. *Journal of Scheduling, 6*, 17–38.

El-Rayes, K., & Kandil, A. (2005). Time–cost–quality trade-off analysis for highway construction. *Journal of Construction Engineering and Management, 131*(4), 477–486.

Erenguc, S. S., Ahn, T., & Conway, D. G. (2001). The resource constrained project scheduling problem with multiple crashable modes: An exact solution method. *Naval Research Logistics, 48*(2), 107–127.

Fallah-Mehdipour, E., Bozorg Haddad, O., Tabari, M. R., & Mariño, M. A. (2012). Extraction of decision alternatives in construction management projects: Application and adaptation of NSGA-II and MOPSO. *Expert Systems with Applications, 39*(3), 2794–2803.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics, 49*(5), 433–448.

Hazır, Ö., Erel, E., & Günalay, Y. (2011). Robust optimization models for the discrete time/cost trade-off problem. *International Journal of Production Economics, 130*(1), 87–95.

Iranmanesh, H., Skandari, M., & Allahverdiloo, M. (2008). Finding Pareto optimal front for the multi-mode time, cost quality trade-off in project scheduling. *World Academy of Science, Engineering and Technology, 38*, 512–516.

Kaplan, L. 1988. Resource-constrained project scheduling with preemption of jobs. Unpublished PhD Dissertation. University of Michigan, Michigan, USA.

Khalili-Damghani, K., Abtahi, A.-R., & Tavana, M. (2013). A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems. *Reliability Engineering and System Safety, 111*, 58–75.

Khalili-Damghani, K., & Amiri, M. (2012). Solving binary-state multi-objective reliability redundancy allocation series-parallel problem using a hybrid efficient epsilon-constraint constraint, multi-start partial bound enumeration algorithm, and DEA. *Reliability Engineering and System Safety, 103*, 35–44.

Khalili-Damghani, K., Tavana, M., & Sadi-Nezhad, S. (2012). An integrated multi-objective framework for solving multi-period project selection problems. *Applied Mathematics and Computations, 219*, 3122–3138.

Khang, D., & Myint, Y. (1999). Time, cost and quality trade-off in project management: A case study. *International Journal of Project Management, 17*, 249–256.

Kim, J., Kang, C., & Hwang, I. (2012). A practical approach to project scheduling: Considering the potential quality loss cost in the time–cost tradeoff problem. *International Journal of Project Management, 30*(2), 264–272.

Lino, M. P. (1997). Planificación de proyectos en diagramas de precedencias. Unpublished PhD Dissertation. Universidad de Valencia, Valencia, Spain.

Mavrotas, G. (2009). Effective implementation of the epsilon-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computations, 213*, 455–465.

Möhring, R. H., Schulz, A. S., Stork, F., & Uetz, M. (2003). Solving project scheduling problems by minimum cut computations. *Management Science, 49*, 330–350.

Pollack-Johnson, B., & Liberatore, M. (2006). Incorporating quality considerations into project time/cost tradeoff analysis and decision making. *IEEE Transactions on Engineering Management, 53*, 534–542.

Prabudha, D., Dunne, E. J., Ghosh, J. B., & Wells, C. E. (1995). The discrete time–cost tradeoff problem revisited. *European Journal of Operational Research, 81*, 225–238.

Rahimi, M., & Iranmanesh, H. (2008). Multi objective particle swarm optimization for a discrete time, cost and quality trade-off problem. *World Applied Sciences Journal, 4*(2), 270–276.

Singh, G., & Ernst, A. T. (2011). Resource constraint scheduling with a fractional shared resource. *Operations Research Letters, 39*, 363–368.

Skutella, M. (1998). Approximation algorithms for the discrete time–cost tradeoff problem. *Mathematics of Operations Research, 23*(4), 909–929.

Sonmez, R., & Bettemir, Ö. H. (2012). A hybrid genetic algorithm for the discrete time–cost trade-off problem. *Expert Systems with Applications, 39*(13), 11428–11434.

Sunde, L., & Lichtenberg, S. (1995). Net-present-value cost/time tradeoff. *International Journal of Project Management, 13*(1), 45–49.

Szmerekovsky, G. J., & Venkateshanb, P. (2012). An integer programming formulation for the project scheduling problem with irregular time–cost tradeoffs. *Computers and Operations Research, 39*(7), 1402–1410.

Tareghian, H. R., & Taheri, S. H. (2006). On the discrete time, cost and quality trade-off problem. *Applied Mathematics and Computation, 181*, 1305–1312.

Tareghian, H. R., & Taheri, S. H. (2007). A solution procedure for the discrete time, cost and quality tradeoff problem using electromagnetic scatter search. *Applied Mathematics and Computation, 190*, 1136–1145.

Van Peteghem, V., & Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research, 201*(2), 409–418.

Vanhoucke, M. (2005). New computational results for the discrete time/cost trade-off problem with time-switch constraints. *European Journal of Operational Research, 165*(2), 359–374.

Vanhoucke, M., Debels, D., & Sched, J. (2007). The discrete time/cost trade-off problem: Extensions and heuristic procedures. *Journal of Scheduling, 10*(4–5), 311–326.

Vanhoucke, M., Demeulemeester, E., & Herroelen, W. (2002). Discrete time/cost trade-offs in project scheduling with time switch constraints. *Journal of the Operational Research Society, 53*, 1–11.

Wuliang, P., & Chengen, W. (2009). A multi-mode resource-constrained discrete time–cost trade-off problem and its genetic algorithm based solution. *International Journal of Project Management, 27*(6), 600–609.

Xu, J., Zheng, H., Zeng, Z., Wu, S., & Shen, M. (2012). Discrete time–cost–environment trade-off problem for large-scale construction systems with s under fuzzy uncertainty and its application to Jinping-II hydroelectric project. *International Journal of Project Management, 30*(8), 950–966.

Yang, T. (2011). Stochastic time–cost tradeoff analysis: A distribution-free approach with focus on correlation and stochastic dominance. *Automation in Construction, 43*, 1–11.

Yang, H., & Chen, Y. (2000). Finding the critical path in an activity network with time-switch constraints. *European Journal of Operational Research, 120*(3), 603–613.

Yu, X., & Gen, M. (2010). *Introduction to evolutionary algorithms*. London: Springer-Verlag.

Zhang, H., & Xing, F. (2010). Fuzzy-multi-objective particle swarm optimization for time–cost–quality tradeoff in construction. *Automation in Construction, 19*(8), 1067–1075.