Optimization Methods & Software, 2015
Vol. 30, No. 5, 965–1001, http://dx.doi.org/10.1080/10556788.2015.1005838

Taylor & Francis
Taylor & Francis Group

Solving multi-mode time–cost–quality trade-off problems under generalized precedence relations

Kaveh Khalili-Damghani[a], Madjid Tavana[b,c*], Amir-Reza Abtahi[d] and Francisco J. Santos Arteaga[e]

[a]*Department of Industrial Engineering, South-Tehran Branch, Islamic Azad University, Tehran, Iran;* [b]*Business Systems and Analytics Department, Lindback Distinguished Chair of Information Systems and Decision Sciences, La Salle University, Philadelphia, PA 19141, USA;* [c]*Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, D-33098 Paderborn, Germany;* [d]*Department of Knowledge Engineering and Decision Sciences, University of Economic Sciences, Tehran, Iran;* [e]*Departamento de Economía Aplicada II, Universidad Complutense de Madrid, Madrid, Spain*

(*Received 20 August 2013; accepted 4 January 2015*)

In this paper, we model a multi-mode time–cost–quality trade-off project scheduling problem under generalized precedence relations using mixed-integer mathematical programming. Several solution procedures, including the classical epsilon-constraint, the efficient epsilon-constraint method, dynamic self-adaptive multi-objective particle swarm optimization (DSAMOPSO), and the multi-start partial bound enumeration algorithm, are provided to solve the proposed model. Several test problems are simulated and solved with the four methods and the performance of the methods are compared according to a set of accuracy and diversity comparison metrics. Additional analyses and tests are performed on the generated Pareto fronts of the solution procedures. Computational experiments are conducted to determine the validity and the efficiency of the DSAMOPSO method. The results show that this method outperforms the other three methods. We also carry out a sensitivity analysis of the DSAMOPSO algorithm to study the effects of parameter changes on the CPU time.

Keywords: time–cost–quality trade-off; generalized precedence relations; particle swarm optimization; partial bound enumeration algorithm; efficient epsilon-constraint method

1. Introduction

Project management is designed to manage or control company resources on a given activity, within time, cost, and quality constraints. Very few projects are completed without incurring trade-offs on time, cost, and quality [22]. A successful project manager needs to keep a balance between the three constraints so that the outcome of the project is not compromised. However, keeping the balance and trade-offs between these three objectives is inherently a difficult task [21]. There are many situations in which the project completion time needs to be compressed due to a variety of unavoidable reasons. Numerous project management methods for schedule compression have been documented in the literature. These methods may include hiring extra staff, investing in more efficient equipment, and/or adopting effective quality control procedures.

*Corresponding author. Email: tavana@lasalle.edu; web: http://tavana.us/

As a result, these methods generally increase the direct cost of the project [52]. Compressing the schedule and balancing the time, cost, and quality constraints are challenging tasks. Yet, the expertise necessary to achieve these goals is limited [2,3]. A detailed review of the multi-mode project scheduling methods can be found in [20,30,50]. The time–cost trade-off problem (TCTP) is a special case that addresses both discrete and continuous time–cost relationships. The TCTP in this paper is limited to the discrete case.

In the TCTPs, there is a trade-off between the cost and time of a project. Solutions with low costs usually take longer while solutions with shorter duration usually cost more. A time, cost, and quality trade-off problem (TCQTP) is an extension of the TCTP. This problem assumes that all project activities can be accomplished in the different modes of cost, time, and quality. The overall goal in TCQTPs is to select each activity's mode such that the project can meet the deadline with the minimum possible cost and the maximum achievable quality. TCQTPs are NP-hard problems because the time, cost, and quality objectives are in conflict with each other.

We propose a new mixed-integer mathematical formulation for solving discrete generalized precedence relations (GPRs) multi-objective multi-mode TCQTPs. Four multi-objective solution procedures, including the classic epsilon-constraint (CEC) method, the efficient epsilon-constraint (EEC) method, dynamic self-adaptive multi-objective particle swarm optimization (DSAMOPSO), and multi-start partial bound enumeration algorithm (MSPBEA), are proposed to solve the proposed model. The proposed procedures generate several sets of non-dominated solutions to assist the decision-makers in defining their preferences for the objective functions. We measure the performance of the proposed procedures using several accuracy and diversity metrics for multi-objective programming. We also analyse and compare the performance of all procedures on a set of systematically generated instances. In summary, the main contributions of this paper in threefold: (1) we study multi-mode time–cost–quality trade-off project scheduling problems under GPRs which is a new problem well-posed to represent real-life projects and not studied in the literature; (2) we propose several exact and heuristic solution procedures to solve these problems; and (3) we use design of experiments (DoE) to tune the parameters in the proposed solution procedure.

The proposed problem fits well with real-life project scheduling problems in which the quality of the activities are as important as the time and cost of the project. In addition, in conventional project scheduling problems, an activity cannot be started unless its preceding activity is completed. This limited assumption is not always true in real-life projects. For instance, a successor may start as the predecessor starts. Moreover, time lags may be defined between starts or finishes of successors and predecessors. The GPR which is modelled in this research, considers all of such situations.

It should be highlighted that there is not a unique optimal solution for the problem, since by considering three conflicting objective functions of quality, time, and cost, one cannot improve an objective without deteriorating at least one of the others. The decision-makers often do not have enough information about the behaviour of these conflicting objectives and therefore cannot provide their preferences on the priority of these objectives. Thus, generating non-dominated solutions on the Pareto front is a desirable strategy for avoiding these trade-offs. In this paper, we propose several multi-objective solution procedures (i.e. CEC, EEC, DSAMOPSO, and MSPBEA) to generate non-dominated solutions. Moreover, the performance of the solution procedures has been compared according to well-known metrics. This approach provides an applicable and executable procedure for solving real-life project scheduling problems.

The remainder of this paper is organized as follows. In Section 2, we present a brief review of the literature in multi-objective optimization and project scheduling. In Section 3, we present the problem formulation and notations. In Section 4, we present the details of all solution procedures, including CEC, EEC, DSAMOPSO, and MSPBEA methods proposed in this study. We discuss

the results and compare the performance of all solution procedures in Section 5 and conclude our paper with a presentation of some future research directions in Section 6.

2. Brief review of literature

Real-life decision-making problems often involve multiple and conflicting objectives where improving an objective may compromise or worsen another one. Therefore, a single solution that optimizes all objectives simultaneously does not generally exist. This type of problems can be modelled with a multi-objective decision-making (MODM) paradigm. MODM problems require decision-makers to choose an element from a set of efficient solutions. That is, decision-makers are expected to consider only the best trade-off, or Pareto optimal, solutions. The Pareto optimality concept was first proposed by Stadler [44] and, when applied to MODM problems, a Pareto optimal solution is defined by Sakawa [40] as follows:

DEFINITION 2.1 *In an MODM problem with minimization form, x^* is said to be a Pareto optimal or Non-dominated solution, if and only if there does not exist another $x \in X$ such that $f_i(x) \leq f_i(x^*)$ for all i and $f_j(x) < f_j(x^*)$ for at least one j.*

The TCQTPs are well-known multi-objective optimization problems that have been studied in continuous and discrete variants. The continuous variant of the problem assumes that there are general continuous functions that relate time, cost, and quality to each other while the discrete variant of the problem assumes that the relationships between time, cost, and quality of a project are defined as discrete points [11]. The multi-mode TCQTPs are NP-Hard problems with a large solution space because each activity can be executed in a particular mode with its own time, cost, and quality value [7,8,11]. Therefore, it is not possible to develop a polynomial time order algorithm for medium and large size instances of the NP-Hard TCQTPs. Heuristic algorithms usually generate suitable solutions (qualified and low computational time) on the Pareto front of the NP-Hard problems. The best known solutions generated by the heuristic and meta-heuristic methods are kept in the project scheduling problem library.[1]

Although different mathematical and exact algorithms have been proposed for small size instances of the TCQTPs, developing exact and heuristic procedure for medium- and large-scale instances is still challenging. Different meta-heuristics have been proposed for solving multi-objective optimization problems. Due to their population-based nature, evolutionary algorithms (EAs) are used to approximate the whole Pareto set (Pareto front) of a multi-objective optimization problem in a single run. There has been a growing interest in applying EAs to deal with multi-objective optimization problems since Schaffer's seminal work [41]. These EAs are called multi-objective evolutionary algorithms (MOEAs). The non-dominated sorting genetic algorithm II (NSGA-II) is a well-known MOEA first introduced by Deb *et al.* [9] (see [53]) for a comprehensive survey of the state of the art of MOEAs). Table 1 presents some related studies in the field of TCQTP.

According to our best knowledge and based on the contents of Table 1, it can be concluded that there is either no or a very limited number of research works in which discrete time–cost–quality trade-off problems (DTCQTPs) in the presence of GPRs are investigated. It should be noted that in real-life projects the relations between the activities of a project are presented in a generalized form, and there are several ways to complete an activity, called multi-mode. Time, cost, and quality values are associated with each implementation mode. Moreover, most of the recent research in the field of DTCQTPs have used meta-heuristic methods which have an efficient solution but are time-consuming and laborious in coding and customization for best performance.

Table 1. Summary of the TCQTP studies.

Author(s)	Method	Main contributions
Babu and Suresh [3]	Linear programming	Using three inter-related linear programming model simply extendable to nonlinear models
Khang and Myint [28]	Linear programming	Applying Babu and Suresh method to an actual cement factory; finding the method's applicability, assumptions and limitations
El-Rayes and Kandil [16]	Genetic algorithm	Applying their model in highway construction projects; quantifying quality by some quality indices; calculating the project quality by addetive weighting of activities quality
Tareghian and Taheri [45]	Integer programming	Developing a method to pruning the activities executions modes
Pollack-Johnson and Liberatore [36]	Goal programming	Conceptualization of quality in projects; Quantifying quality of each activity execution mode by AHP; Developing a goal programming model with four objectives, namely time, cost, minimmum quality, and mean of quality
Tareghian and Taheri [46]	Electromagnetic scatter search	Validating and cheking the applicability of their algorithm by solving a randomly generated large-scale problem with 19,900 activities
Afshar *et al.* [1]	Multi-colony ant algorithm	Solving an example and comparing their algorithm's results with some other algorithms
Rahimi and Iranmanesh [38]	MOPSO	Every activity had several different modes offered by various options of time, cost, and quality and the best options of the project's activities were determined to minimize the total cost of the project while maximizing the quality and also meeting a given deadline by assuming that duration and quality of project activities are discrete
Shahsavari Pour *et al.* [42]	Hybrid genetic algorithm	Proposing a mathematical formulation for the multi-mode time–cost–quality trade-off problem; solving the problem using a new hybrid genetic algorithm
Zhang and Xing [52]	Particle swarm optimization	Considering construction methods instead of execution modes for each activity; using fuzzy numbers to describe time, cost, and quality; fuzzy muti-attribute utility methodology and constrained fuzzy arthmetic operators to evaluate each construction method; demonstrating the effectiveness of their algorithm by solving a bridge construction project TCQT problem
Kim *et al.* [29]	Mixed-integer linear programming	Focusing on minimizing quality loss cost instead of maximizing the individual acticity quality of projects; validating their model by applying it to a robot type palletizing system installation project

However considering the technological advances of emergent CPUs, efficient exact procedures can effectively handle the combinatorial optimization problems.

In this research, we propose a mixed-integer mathematical programming model for the DTC-QTP in the presence of multi-mode for activities and GPRs. Moreover, an efficient solution procedure based on the epsilon-constraint method is also proposed to solve several instances of the proposed problem. The proposed solution procedure, called the EEC, generates several sets of non-dominated solutions on the Pareto front of instances of the problem.

It is notable that the EEC method has proven to be robust for complicated real-life multi-objective mathematical programming problems [23–27,48].

3. Problem formulation and notations

Consider a project 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) type of the project representation. Each activity $i \in V$, can be executed in $r(i)\ \in k$ different modes and each mode k has its own time (t_{ik}), cost (c_{ik}), and quality (q_{ik}).

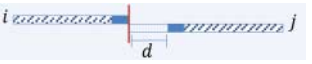
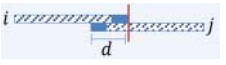
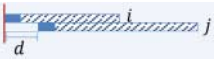
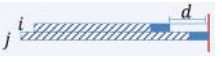
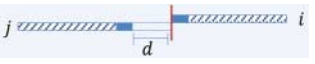
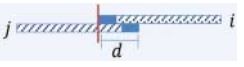
3.1 *Generalized precedence relations*

Real-life project scheduling problems involve several types of precedence relations among the project activities. As the GPR can handle real-life project scheduling problems more effectively, several research streams have been dedicated to this area in recent years. De Reyck and Herroelen [13] first proposed a branch-and-bound procedure for the resource-constrained project scheduling problem (RCPSP) with GPRs. De Reyck and Herroelen [14] proposed the RCPSP with discounted cash flows and generalized precedence relations (RCPSPDC-GPR). De Reyck and Herroelen [15] tackled the challenging problem of scheduling activities to minimize the project duration, in which the activities (a) were subject to GPRs, (b) required units of multiple renewable, non-renewable and doubly constrained resources for which a limited availability is imposed, and (c) could be performed in one of several different ways, reflected in multiple activity scenarios or modes.

Herroelen *et al.* [19] reviewed the RCPSPs. They classified optimal solution procedures for the basic problem and subsequently illustrated extensions to a rich and realistic variety of related problems involving activity preemption, the use of ready times and deadlines, variable resource requirements and availabilities, GPRs, time/cost, time/resource and resource/resource trade-offs, and non-regular objective functions. Brucker *et al.* [5] reviewed the notations, classes, models, and solution procedures for RCPSPs. They classified time lags and GPRs. Brucker and Knust [6] presented a destructive lower bound for the multi-mode RCPSPs with minimal and maximal time lags in the form of GPRs. They reported computational results for several test instances on the multi-mode problem with and without time lags and the single-mode version with time lags. Neumann *et al.* [35] proposed a multi-mode problem where for each GPR between activity i and j, the associated minimal or maximal time lag depended on the execution modes of both activities i and j. They presented the set of time lags associated with a given GPR by means of a matrix.

Najafi *et al.* [34] associated a resource investment problem with discounted cash flows to a project scheduling problem with GPRs. They proposed a genetic algorithm to solve the model. Hartmann and Briskorn [18] presented a survey of variants and extensions of the RCPSP. They also described popular variants and extensions, such as multiple modes, minimal and maximal time lags, and net present value-based objectives. Kyriakidis *et al.* [32] proposed new

Table 2. Generalized precedence relations.

Notation	Time lag	Graph
FS	Zero	
FS	Positive	
FS	Negative	
SS	Zero	
SS	Positive	
SS	Negative	
FF	Zero	
FF	Positive	
FF	Negative	
SF	Zero	
SF	Positive	
SF	Negative	

mixed-integer linear programming models for the deterministic single- and multi-mode RCPSP with renewable and non-renewable resources. They considered several scenarios and then efficiently transformed them into mathematical formulations including a set of constraints describing precedence relations and different types of resources.

Bianco and Caramia [4] proposed a new lower bound for the RCPSP with generalized precedence relationships. Quintanilla *et al.* [37] presented an application of project scheduling concepts and solution procedures that was used to resolve a complex problem that comes up in the daily management of many company service centres. The real problem has been modelled as a multi-mode RCPSP with preemption, time and work generalized precedence relationships with minimal and maximal time lags between the tasks and due dates. Amiri *et al.* [2] solved a generalized precedence multi-objective multi-mode time–cost–quality trade-off project scheduling problem using a modified NSGA-II algorithm. Tavana *et al.* [47] proposed a new multi-objective multi-mode model for solving DTCQTPs with preemption and GPRs.

The relations between two activities can be finish-to-start (FS), start-to-start (SS), finish-to-finish (FF), or start-to-finish (SF). In some cases, there are time lags (leads) between project activities. For example, between two activities i and j with an FS precedence relation, activity j has to start after finishing activity i and an elapsing time lag (i.e. FS + 1). Table 2 demonstrates all possible forms of general precedence relations (GPR) between activities in a project.

In order to consider GPRs, the set of arcs, E, is partitioned into four subsets denoted E_{SS}, E_{SF}, E_{FS}, and E_{FF}, which, respectively, denote the sets of SS, SF, FS, and FF precedence relations.

Table 3. Problem formulation notations.

Counter	Statement	Range
i	Counter for activity number	$i = 1, 2, \ldots, n$
j	Counter for activity number	$j = 1, 2, \ldots, n$
k	Counter for activity mode	$k = 1, 2, \ldots, r(i)$
V	Set of all activities in project	
Parameter		
n	Number of project activities	
C	Upper bound of project cost	
T	Project deadline	
Q	Minimum desired level of quality of project	
r_i	Number of execution modes for activity i	$i = 1, 2, \ldots, n$
t_{ik}	Time of activity i in mode k	$i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, r(i)$
c_{ik}	Cost of activity i in mode k	$i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, r(i)$
q_{ik}	Quality of activity i in mode k	$i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, r(i)$
d_{ij}	Time lag (lead) between activity i and activity j	$i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n$
w_i	The quality weight for activity i	$i = 1, 2, \ldots, n$
Decision variable		
x_{ik}	Equals 1 if activity i executes in mode k and is 0, otherwise	$i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, r(i)$
s_i	Start time of activity i	$i = 1, 2, \ldots, n$

3.2 *Problem formulation*

The objectives faced by a decision-maker are to minimize the overall time and cost and maximize the overall quality of the project. The problem is then to select a mode for each project activity and generate a set of non-dominated solutions. The notations used to formulate the discrete generalized precedence TCQTP are presented in Table 3.

$$\min \ \text{Time} = S_n, \tag{1}$$

$$\text{Min Cost} = \sum_{i=1}^{n} \sum_{k=1}^{r_i} x_{ik} c_{ik}, \tag{2}$$

$$\text{Max Quality} = \frac{\left(\sum_{i=1}^{n} \sum_{k=1}^{r_i} w_i x_{ik} q_{ik}\right)}{\sum_{i=1}^{n} w_i} \tag{3}$$

s.t.

$$\sum_{k=1}^{r_i} x_{ik} = 1 \quad \forall\, i \in V, \tag{4}$$

$$s_i + \sum_{k=1}^{r_i} t_{ik} x_{ik} + d_{ij} \le s_j \quad \forall\, i,j \in E_{\text{FS}}, \tag{5}$$

$$s_i + d_{ij} \le s_j + \sum_{k=1}^{r_i} t_{jk} x_{jk} \quad \forall\, i,j \in E_{\text{SF}}, \tag{6}$$

$$s_i + d_{ij} \le s_j \quad \forall\, i,j \in E_{\text{SS}}, \tag{7}$$

$$s_i + \sum_{k=1}^{r_i} t_{ik} x_{ik} + d_{ij} \le s_j + \sum_{k=1}^{r_i} t_{jk} x_{jk} \quad \forall\, i,j \in E_{\text{FF}}, \tag{8}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{r_i} x_{ik}c_{ik} \le C, \tag{9}$$

$$s_n \le T, \tag{10}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{r_i} x_{ik}q_{ik} \ge Q, \tag{11}$$

$$s_i \ge 0 \quad \forall\, i \in V, \tag{12}$$

$$x_{ik} \in \{0,1\} \quad \forall i,k, \tag{13}$$

Relations (1)–(3) are allocated to describe time, cost, and quality objective functions. Constraint (4) guarantees the selection of one and only one mode for each activity. Constraints (5)–(8) preserve the GPRs between project activities. Constraint (9) restricts the upper bound for the cost of the project. Constraint (10) restricts the upper bound for the time of the project. Constraint (11) assures the minimum level of quality for the generated schedules. Constraint (12) ensures that no activity can start earlier than zero. Constraint (13) represents the binary form of the decision variables.

It should be mentioned that the constraints (9)–(11) restrict the solution within a specific region of the Pareto front in which the time, cost, and quality of the project satisfy the predetermined lower and upper bounds. The upper bound of constraints (9)–(10) and lower bound of constraint (11) are determined by the decision-maker as parameters of the problem. In this way, the non-dominated solutions are pruned and the input of the decision-maker facilitates the search for preferred solutions instead of non-dominated ones. The preferred solutions in this paper are non-dominated solutions on the Pareto front of the problem which are restricted by the lower and upper bounds of the objective functions determined by the decision-maker. On the other hand, the constraints (9)–(11) cause a more efficient search in the plane of the Pareto front. Figure 1 illustrates the concept of non-dominated and preferred solutions for two given objective functions which should be minimized.

Model (1)–(13) is a mixed-integer multi-objective mathematical programming. It pessimistically contains $0.5 \times n^2 + n + 2$ constraints and optimistically contains $2 \times n + 1$ constraints. Model (1)–(13) has $n \times k$ decision variables. It is worthwhile to mention that the optimistic and pessimistic conditions for the number of constraints of Model (1)–(13) are incurred based on the precedence relations of the activities in the project. Moreover, it should be mentioned that the

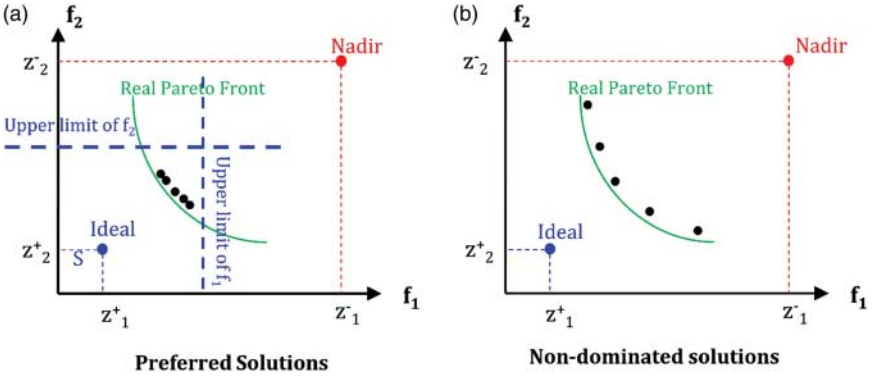


Figure 1. The effect of limits of objective functions on preferred solutions.

solution procedure involves three conflicting objective functions in order to rank and select non-dominated solutions. The simplest ranking procedures impose high complexity to the solution procedure.

According to Demeulemeesterl *et al.* [10], changes from one mode to another will affect the cost of executing the activity as a single resource of the project does in a typical TCTP problem. It is noteworthy that in a typical TCTP problem the cost of all resources are calculated and treated as a non-renewable single resource. Thus, the duration of the activity is a discrete non-increasing function of the single resource, financial capability [10].

General and standard representations of project networks may be used alternatively. For a project with GPRs (i.e. FF, SF, SS, and FS), all types of relations can be converted to SS relations in order to make a standard network. Therefore there is no need for using all the different kinds of constraints but just the SS one. In this case, the computational effort and number of constraints would remain the same but the amount of notations will decrease.

3.3 *Resolving cycle in GPR networks*

A cycle in the project network is a direct effect of defining incorrect precedence relations among activities. For example, if activity A is a predecessor of activity B, activity B is a predecessor of activity C, and activity C is a predecessor of activity A, then, a cycle is formed. In such situations, a schedule cannot be programmed since the feasible space of the problem is empty. In contrast, a well-defined real-life project network should not contain any cycle(s). That said, we suggest using the cycle detection procedure proposed by Ranjbar *et al.* [39] prior to the implementation of the project scheduling method proposed in this study to eliminate infeasible schedules. Nevertheless, we have introduced a mechanism in our proposed model to prevent infeasible schedules.

An infeasible schedule may be the result of wrongly defined precedence relations among activities. In the proposed model (1)–(13), the set of constraints (5)–(8) which control the GPRs between activities, prevent any two given activities from violating the precedence relations. Therefore, if the precedence relations are defined incorrectly and a cycle is generated in the project network, then, constraints (5)–(8) results in infeasibility of model (1)–(13). All proposed algorithms, including exact methods and heuristic ones, have been developed based on model (1)–(13) and include a constraint-handling mechanism.

In special cases, some difficulties may arise in GPR networks. In this type of networks, the relations between activities can be of the minimum or the maximum type. In the maximum relation, the successor activity should be started no later than a predetermined time while in the minimum relation, the successor activity cannot be started earlier than a predetermined time. Consider two given activities A and B, where activity A has a maximum SS relation with activity B with a maximum time lag. Moreover, activity A has also a minimum SS relation with activity B with a minimum time lag. Figure 2 illustrates this situation.

In Figure 2, $SS^{max}_{A,B}$ is the maximum type SS relation between activities A and B, $l^{max}_{A,B}$is the maximum feasible time-lag between starts of activities A and B, $SS^{min}_{A,B}$ is the minimum type SS relation between activities A and B, $l^{min}_{A,B}$ is the minimum feasible time-lag between starts of activities A and B, S^{min}_B is the start time of activity B considering its minimum relation with activity A, and S^{max}_B is the start time of activity B considering its maximum relation with activity A.

Note that, given the maximum relation between both activities, activity B should be started no later than $S^{max}_B = S_A + l^{max}_{A,B}$, while given the minimum relation, activity B cannot be started earlier than $S^{min}_B = S_A + l^{min}_{A,B}$. Both relations will generate a feasible region for the start and finish of the successor activity (i.e. activity B). This feasible region (i.e. $[S_A + l^{min}_{A,B}, S_A + l^{max}_{A,B}]$ is called the time window. A generalized precedence network cannot be efficiently analysed in the presence

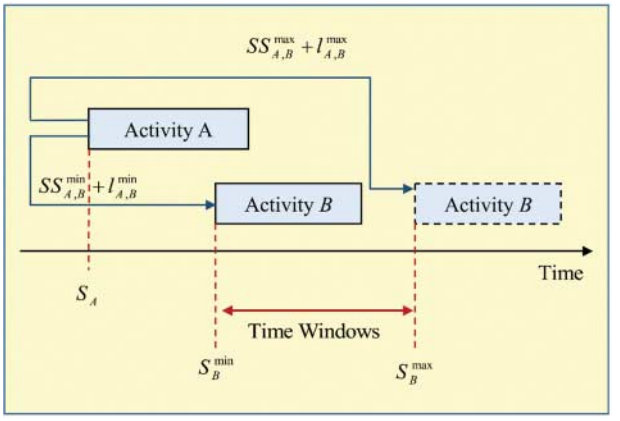


Figure 2. Time windows in the generalized precedence network.

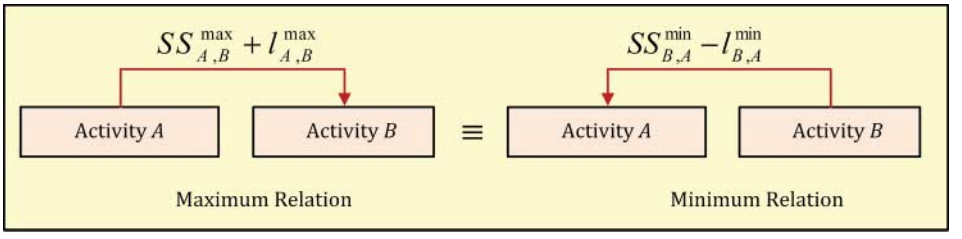


Figure 3. Equivalence of maximum relation.

of both maximum and minimum relations. The existence of a time window will result in several calculation difficulties. We will eliminate the time window using equivalence relations. In order to do so, we must transform both the maximum and minimum relations in a unique relation. Note that, a maximum relation between two given activities, A and B, can be changed into a minimum relation using the following relation:

$$l^{\max}_{A,B} = -l^{\min}_{B,A}.$$

Figure 3 presents the equivalence between a maximum relation and a minimum one.

The equivalence between the maximum and minimum relations described in Figure 3 can be illustrated formally. In order to do so, we must define both relations algebraically using the notations employed in the figure.

(a) $S_A + l^{\max}_{A,B} \geq S_B$,
(b) $S_B - l^{\min}_{B,A} \leq S_A$.

Note that the latter inequality can be rewritten as

(b') $S_A + l^{\min}_{B,A} \geq S_B$,

Thus, in order for (b') and (a) to be equivalent, we require the time lags $l^{\max}_{A,B}$ and $l^{\min}_{B,A}$ to account for the same exact amount of time. In other words, the maximum feasible time lag added between the start times of activities A and B must be identical to the minimum feasible time lag subtracted between the start times of activities B and A.

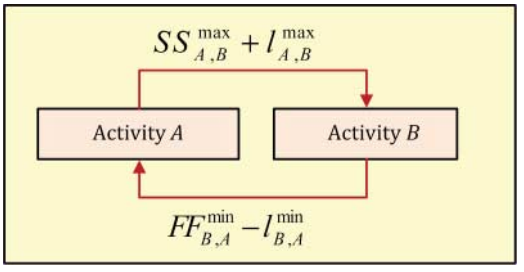


Figure 4. A typical network with cycle.

Trivially, this equivalence relation holds when considering the example presented in Figure 4, where we will make use of the following equivalence:

$$FF^{\max}_{A,B} + l^{\max}_{A,B} \equiv FF^{\min}_{B,A} - l^{\min}_{B,A}. \tag{14}$$

Consider the example presented in Figure 4, which illustrates the formation of a cycle in the network. Note that the $\mathrm{SS}^{\max}_{A,B} + l^{\max}_{A,B}$ relation between activities A and B can be written as $S_A + l^{\max}_{A,B} \geq S_B$, while the $\mathrm{FF}^{\min}_{B,A} - l^{\min}_{B,A}$ relation between activities B and A can be written as follows:

$$F_B - l^{\min}_{B,A} \leq F_A. \tag{15}$$

Note that, in this case, the start time of activity A is given a time window that may displace the start time of activity B, which, at the same time, may displace the finish time of activity A, which, at the same time, may modify the start time of activity A, which, at the same time, may displace the start time of activity B and so on, leading to a cycle between both activities.

After applying the equivalence relation described in Equation (14), we are able to eliminate the cycle by introducing the following relation between activities A and B in place of Equation (15):

$$F_A + l^{\max}_{A,B} \geq F_B$$

Note that both maximum relations between activities have eliminated the cycle that might have arisen when maximum and minimum relations between activities were allowed for in the network.

Note also that Equations (7) and (8), which account for SS and FF relations in our model, respectively, are both initially defined in terms of minimum-type relations between activities i and j. The same is true for Equations (5) and (6) when dealing with FS and SF relations, respectively.

Thus, model (1)–(13) requires identical types of relations among all activities, which prevents the generation of cycles within the network. Moreover, as stated at the beginning of this section, constraints (5)–(8) would lead to infeasible models if cycles of the type:

$$S_A + d_{A,B} \leq S_B + d_{B,C} \leq S_C + d_{C,A} \leq S_A,$$

where, for example, introduced in the set of SS relations defined by Equation (7).

Both measures, the equivalence between maximum and minimum relations together with the design of the constraints of the model, have been implemented to eliminate cycles from the networks being analysed.

4. Solution procedures

In this section, a classical multi-objective mathematical programming, an EEC method, and a multi-objective particle swarm optimization (MOPSO) algorithm are proposed to generate different sets of non-dominated solutions for the discrete generalized precedence TCQTP defined in (1)–(13).

4.1 *Classical epsilon-constraint (CEC) method*

There are several methods that are capable of producing the entire set of efficient solutions for the MODM problems, including linear and mixed-integer programming. These methods can provide a representative subset of the non-dominated solutions. In the CEC method, the decision-maker chooses one objective out of n to be optimized; the remaining objectives are constrained to be less than or equal to given target values. In mathematical terms, assuming that $f_j(x), j \in \{1, \ldots, k\}$ is the objective function chosen to be optimized, we have the following problem:

$$\begin{aligned} &\min\ f_j(x), j \in \{1, \ldots, k\} \\ &\text{s.t.} \\ &f_i(x) \le \varepsilon_i, \quad \forall i \in \{1, \ldots, k\}, \quad i \ne j, \\ &x \in S, \end{aligned} \tag{16}$$

where S is the feasible solution space.

One advantage of the epsilon-constraint method is its ability to achieve efficient points in a non-convex Pareto curve. As a result, the decision-maker can modify the upper bounds ε_i to obtain weak Pareto optima. However, this is also a drawback of this method since the decision-maker also has to choose appropriate upper bounds for the ε_i values. Moreover, the method is not particularly efficient if there are increasing numbers of objective functions in the problem.

It is worthwhile to mention that as a multi-objective mathematical programming is changed into a single-objective mathematical programming, the computational effort of the solution procedure will decrease proportionally since there is no need to use non-dominant sorting procedures in order to achieve the non-dominated solutions.

Several methods have been proposed for improving the epsilon-constraint method [33]. More formally, the epsilon-constraint method has three points that need attention in its implementation: (a) the calculation of the range of the objective functions over the efficient set, (b) the guarantee of efficiency of the obtained solution and, (c) the increased solution time for problems with several objective functions. Mavrotas [33] tried to address these issues with an efficient version of the epsilon-constraint method.

In this regard, the EEC method has been successfully utilized in several settings. The success and details of this method can be found in the literature [23–27,33,48]. We briefly introduce the mechanics of this method and skip a discussion of its applications for the sake of brevity.

4.2 *EEC method for discrete generalized precedence TCQTPs*

The customization result of the EEC method proposed by Mavrotas [33] for model (1)–(13), is presented as relations (15)–(18):

$$\text{Min Time} - \beta \times \left(\frac{S_2}{r_2} + \frac{S_3}{r_3} \right) \tag{17}$$

s.t.

$$\text{Cost} + S_2 = \varepsilon_2, \quad \varepsilon_2 \in [\text{Cost}^-, \text{Cost}^+], \tag{18}$$

$$\text{Quality} - S_3 = \varepsilon_3, \quad \varepsilon_3 \in [\text{Quality}^-, \text{Quality}^+], \tag{19}$$

$$X \in S, \tag{20}$$

where r_2 and r_3 represent the range of the cost and quality objective functions, respectively. The range of each objective is calculated by using the ideal and nadir values of the cost and quality in the lexicographic payoff table of the original discrete generalized precedence TCQTP. The *cost* and *quality* variables have the same definitions as in Model (4)–(13). Cost^- and Cost^+ are the nadir and ideal values of the single cost optimization problems (4)–(13), respectively. Similarly, Quality^- and Quality^+ are the nadir and ideal values of the single quality optimization problems (4)–(13), respectively. $X \in S$ is the feasible region of Model (3)–(14) and β is a small positive value used to control the effect of the term $(S_2/r_2 + S_3/r_3)$ in the objective function.

It is notable that the nadir and ideal values are calculated based on payoff tables. In order to achieve the nadir value of a maximization problem, other objective functions are ignored and the associated single-objective minimization problem of the original MODM problem is solved optimally. In order to achieve the nadir value of a minimization problem, other objective functions are ignored and the associated single-objective maximization problem of the original MODM problem is solved optimally. The ideal values of each objective function are calculated as follows. In order to achieve the ideal value of a maximization problem, other objective functions are ignored and the associated single-objective maximization problem of the original MODM problem is solved optimally. In order to achieve the ideal value of a minimization problem, other objective functions are ignored and the associated single-objective minimization problem of the original MODM problem is solved optimally.

The EEC method has the following two advantages over the classical epsilon-constraint (CEC) method: (1) the calculation of the range of the objective functions over the efficient set is accomplished based on the lexicographic payoff table, and (2) the process of searching the solution space is done systematically to avoid redundant calculations and to decrease the solution time for the discrete generalized precedence TCQTP with more than two objective functions.

The values of the slack variables in the second and the third objective functions (i.e. S_2 and S_3) are different because the scales of these two objective functions are dissimilar and should be put into a common scale. In this regard, $r_i,\ i = 2, 3$ helps resolving the problem of different scales in the conflicting objective functions.

The term $(S_2/r_2 + S_3/r_3)$ helps generating possible strong non-dominated solutions on the Pareto front in Model (4)–(13). In other words, when the objective function in Model (17)–(20) is to be minimized, $(S_2/r_2 + S_3/r_3)$ should be maximized. Note that $r_i,\ i = 1, 2$ are the ranges of the objective functions and, therefore, fixed values. Hence, the term $(S_2/r_2 + S_3/r_3)$ can be maximized if and only if the slack variables of the second and the third objective functions are maximized. Considering the associated constraints in Model (17)–(20) this is a good way to generate strong non-dominated solutions (lower cost and higher quality while minimizing time) via changing ε_2 and ε_3.

4.3 *Dynamic self-adaptive multi-objective particle swarm optimization (DSAMOPSO)*

We have customized the DSAMOPSO method proposed by Khalili-Damghani *et al.* [23] in order to solve discrete generalized precedence TCQTPs. In this section, we discuss the basics of the DSAMOPSO method. Details can be found in [23].

4.3.1 *Mathematics of classic particle swarm optimization (PSO)*

Let us consider a swarm which includes m particles that are seeking the optimum value of the objective function(s) in an n-dimensional search space, each particle having a vector of position $\vec{X}_i = (x_{i1}, x_{i2}, \ldots, x_{in}), \quad i = 1, 2, \ldots, m$ which is associated with a solution, a vector of velocity $\vec{V}_i = (v_{i1}, v_{i2}, \ldots, v_{in}), \quad i = 1, 2, \ldots, m$ which determines the movement value of a particle in each dimension to improve its current position, and a vector of particle best position $\vec{P}_i = (p_{i1}, p_{i2}, \ldots, p_{in}), \quad i = 1, 2, \ldots, m$ which is associated with the most fitted positions of a particle from the first step of the algorithm. It is notable that the fitness of a position can easily be calculated considering the objective function of the optimization problem. A vector of the global best particle $\vec{P}_g = (p_{g1}, p_{g2}, \ldots, p_{gn})$ is reserved for knowledge sharing among all the particles of a swarm. Using the aforementioned notations, each argument of the velocity and position vectors for each particle in the swarm is updated through the multiple iterations of the algorithm using the following model:

$$\begin{aligned} &\vec{V}_i(t+1) = W(t) \times \vec{V}_i(t) + C_1 r_1(\vec{P}_i - \bar{X}_i(t)) + C_2 r_2(\vec{P}_g - \bar{X}_i(t)), \quad i = 1, 2, \ldots, m, \\ &\bar{X}_i(t+1) = \bar{X}_i(t) + \vec{V}_i(t+1), \quad i = 1, 2, \ldots, m, \end{aligned} \tag{21}$$

where W is the inertia weight and determines the tendency of a particle to maintain its previous exploration direction [43], C_1 and C_2 are cognitive and social factors, respectively; $r_1, r_2 \in [0, 1]$ are the random numbers, and t represents the iteration number.

4.3.2 *Structure of a particle in the DSAMOPSO method for the TCQTP*

A binary-coded structure is supplied to depict a particle in the proposed DSAMOPSO method. The structure is presented in Figure 5

where each allele x_{ik} in the particle is either zero or one. It is equal to one if the activity i is accomplished in mode k and it is equal to zero otherwise, and $\sum_{k=1}^{r(i)} x_{ik} = 1, \quad i = 1, 2, \ldots, n$ is preserved for all activities, with $|r(i)|$ referring to the cardinality, that is, number of elements, of the set $r(i)$. It is also notable that each allele x_{ik} in the particle has the same definition as x_{ik} in the mathematical formulation of the *TCQTP* in model (1)–(13).

4.3.3 *Updating position and velocity vector of the DSAMOPSO*

Re-considering Equation (21), the position and the velocity vectors of a particle are, respectively, defined as follows:

$$\begin{aligned} &\vec{V}_i(t+1) = W(t) \times \vec{V}_i(t) + C_1(t) r_1(\vec{P}_i - \bar{X}_i(t)) + C_2(t) r_2(\vec{P}_g - \bar{X}_i(t)), \quad i = 1, 2, \ldots, m, \\ &\bar{X}_i(t+1) = \bar{X}_i(t) + \vec{V}_i(t+1), \quad i = 1, 2, \ldots, m. \end{aligned} \tag{22}$$

The parameters of (22) have the same definitions as the parameters in (21). $W(t)$, $C_1(t)$, and $C_2(t)$ are determined using an iteration function. The random numbers r_1 and r_2 in (22) are generated independently for each particle.

Activity 1				Activity 2				...	Activity *n*			
mode 1	mode 2	...	mode k	mode 1	mode 2	...	mode k	...	mode 1	mode 2	...	mode k
X_{11}	X_{12}	...	X_{1k}	X_{21}	X_{22}	...	X_{2k}	...	X_{n1}	X_{n2}	...	X_{nk}

Figure 5. Structure of a particle in swarm.

4.3.4 *Dynamic inertia weight, cognitive and social factors*

We introduce time variant acceleration coefficients in order to improve the compromise between the exploration and exploitation phases in the PSO.

The value of the inertia weight is determined dynamically by considering the algorithm iteration numbers between W_1 and $W_2 < W_1$ as follows [49]:

$$W(t) = (W_1 - W_2)\frac{\text{Max } t - t}{\text{Max } t} + W_2, \tag{23}$$

where W_1 and W_2 are two parameters, Max t is the maximum allowed number of iterations and t is the current iteration number. The best values for W_1 and W_2 are experimentally determined as suggested by Tripathi *et al.* [49].

The values of the cognitive and the social factors are also determined dynamically by considering the algorithm iteration number as follows [49]:

$$C_1(t) = (C_{1\text{f}} - C_{1\text{i}})\frac{t}{\text{Max } t} + C_{1\text{i}}, \tag{24}$$

$$C_2(t) = (C_{2\text{f}} - C_{2\text{i}})\frac{t}{\text{Max } t} + C_{2\text{i}}, \tag{25}$$

C_1 is allowed to decrease from an initial value, called $C_{1\text{i}}$, to a final value, denoted $C_{1\text{f}}$. C_2 is increased from an initial value, called $C_{2\text{i}}$, to a final value, denoted $C_{2\text{f}}$. The best values of $C_{1\text{i}}$, $C_{1\text{f}}$, $C_{2\text{i}}$, and $C_{2\text{f}}$ are experimentally determined in a similar way to Tripathi *et al.* [49]. We should note that Max t and t have the same definition as in (23).

4.3.5 *Handling the constraints in the proposed DSAMOPSO*

A penalty strategy is considered in the DSAMOPSO method in order to implement the constraints of Model (1)–(13). The constraints (4)–(11) in the *TCQTP* are implemented using a dynamic self-adaptive penalty function. In this strategy, the iteration of the algorithm and the overall situation of the swarm are concurrently considered. The violation value of each particle in the swarm is calculated considering all constraints as $v_{ic}, i = 1, \ldots, m;\ c = 4, 5, \ldots, 11$. Then the total violation of a particle i is calculated as $v_i = \sum_{c=4}^{11} v_{ic},\ i = 1, 2, \ldots, m$.

The dynamic self-adaptive penalty functions for the time, cost, and the quality objectives are defined as follows:

$$\text{Time}'_i = \text{Time}_i + \left[\left(\frac{v_i}{v_i^{\min}}\right)^{\alpha} \times t^{\beta}\right], \tag{26}$$

$$\text{Cost}'_i = \text{Cost}_i + \left[\left(\frac{v_i}{v_i^{\min}}\right)^{\alpha} \times t^{\beta}\right], \tag{27}$$

$$\text{Quality}'_i = \text{Quality}_i - \left[\left(\frac{v_i}{v_i^{\min}}\right)^{\alpha} \times t^{\beta}\right], \tag{28}$$

where Time'_i, Cost'_i, and $\text{Quality}'_i$ are the modified values of time, cost, and quality objectives for the violated particle i in the swarm; v_i is the total violation of a particle i, $v_i^{\min} = \min_i\{\varepsilon + v_i\}$ is the minimum violation value in the swarm; ε is a small positive value used to avoid division by zero; t is the iteration number; and α, β are the control parameters which are experimentally determined.

This type of dynamic self-adaptive penalty function allows for a more efficient search of the solution space and guarantees that the violations in the final steps of the algorithm occur at a

greater cost. Moreover, the penalty value has been calculated using the minimum violation of the swarm. This latter property facilitates finding the best particles in the swarm.

4.4 *Proposed MSPBEA*

We have customized the idea of PBEA first proposed by Khalili-Damghani and Amiri [25] in order to handle the TCQTP. The idea of PBEA was first proposed by Khalili-Damghani and Amiri [25] for integer programming. We have modified it completely in order to handle TCQTP in which the main decision variables are binary. Consider a typical binary multi-objective optimization problem as Model (29).

$$\begin{aligned} &\text{Maximize} \quad (f_1, f_2, \ldots, f_p) \\ &\text{s.t.} \\ &g_j(x) \le b_j, \quad j = 1, 2, \ldots, m, \\ &x_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{29}$$

where $f_1, f_2, \ldots, f_p$ are objectives of the problem, $g_j(x) \le b_j, \quad j = 1, 2, \ldots, m$ are the set of constraints of the problem, and $x_i \in \{0, 1\} \quad i = 1, 2, \ldots, n$ are binary decision variables. The steps of the proposed multi-start PBEA for TCQTP can be summarized as follows:

Step 1 Initialize semi-random solutions. Since the main decision variables of the TCQTP are binary, set x_{ik}^{L} and x_{ik}^{U}, the lower bound and the upper bound of the decision variables, equal to 0 and 1, respectively. For each activity i randomly select a mode k and set the associated x_{ik} equal to 1. Generate P semi-random solutions $X_1, X_2, \ldots, X_p$ where P is a parameter of the algorithm and $X_p = (x_{11}^p, x_{12}^p, \ldots, x_{1k}^p, \ldots, x_{i1}^p, x_{i2}^p, \ldots, x_{ik}^p, \ldots, x_{n1}^p, x_{n1}^p, \ldots, x_{nk}^p)$.

Step 2 Find feasible solution. Considering the m different constraints of Model (29), resolve the infeasibility of a solution X_p as follows. Choose one of the n blocks of an infeasible solution randomly. A block has k elements and is associated with an activity in solution P. Assume, for example, block i in solution X_p is $x_{i1}^p, x_{i2}^p, \ldots, x_{ik}^p$, $i = 1, 2, \ldots, n$. Find the element of the selected block which is equal to 1 and set this element to 0. Select another element in the block randomly and set it to 1. Check the feasibility of the solution subject to the m different constraints of Model (29). If the violation of the new solution is less than the violation of the old solution accept the change, otherwise try another random element in the block. Continue until the minimum violation value is reached through the changes implemented in this block and no further improvement can be obtained. If the minimum violation value is equal to zero a feasible solution has been achieved; stop and save the feasible solution. Otherwise, select another random block of the infeasible solution and try to decrease the violation of the whole solution using the above approach in a new randomly selected block.

Step 3 Find bound solution. In a feasible solution, select block i randomly. Find the element of the selected block which is equal to 1 and set this element to 0. Select another element in the block randomly and set it to 1. Check the feasibility of the solution subject to the m different constraints of Model (29). If more than one constraint is violated discard the last change. Repeat this using different randomly selected blocks until at least one constraint is violated; then discard the last change. The solution is a proper estimation of a bound solution for the problem.

Step 4 Systematic search of solution space. Change the structure of a bound solution in such a way that the combination of objective functions is improved in the sense of generating a non-dominated solution. Stop if there is no probability of generating a new non-dominated solution considering feasibility conditions. Put all generated non-dominated solutions in the temporary archive of non-dominated solutions.

Step 5 Choose the final non-dominated solutions. Rank all available solutions in the temporary

archive of non-dominated solutions according to non-dominated sorting. If the final archive of non-dominated solutions is empty, then put all the solutions which have taken rank 1 into the final archive of non-dominated solutions. Otherwise, add only the non-dominated solutions extracted from the union of existing solutions in the final archive of non-dominated solutions and the temporary archive of non-dominated solutions. Empty the temporary archive of non-dominated solutions. If the termination conditions are met, print the set of existing solutions in the final archive of non-dominated solutions; otherwise go to step 1.

Using the aforementioned logic, there is no need to check the whole solution space of a binary optimization problem. Only those feasible solutions which are on the boundary of the solution

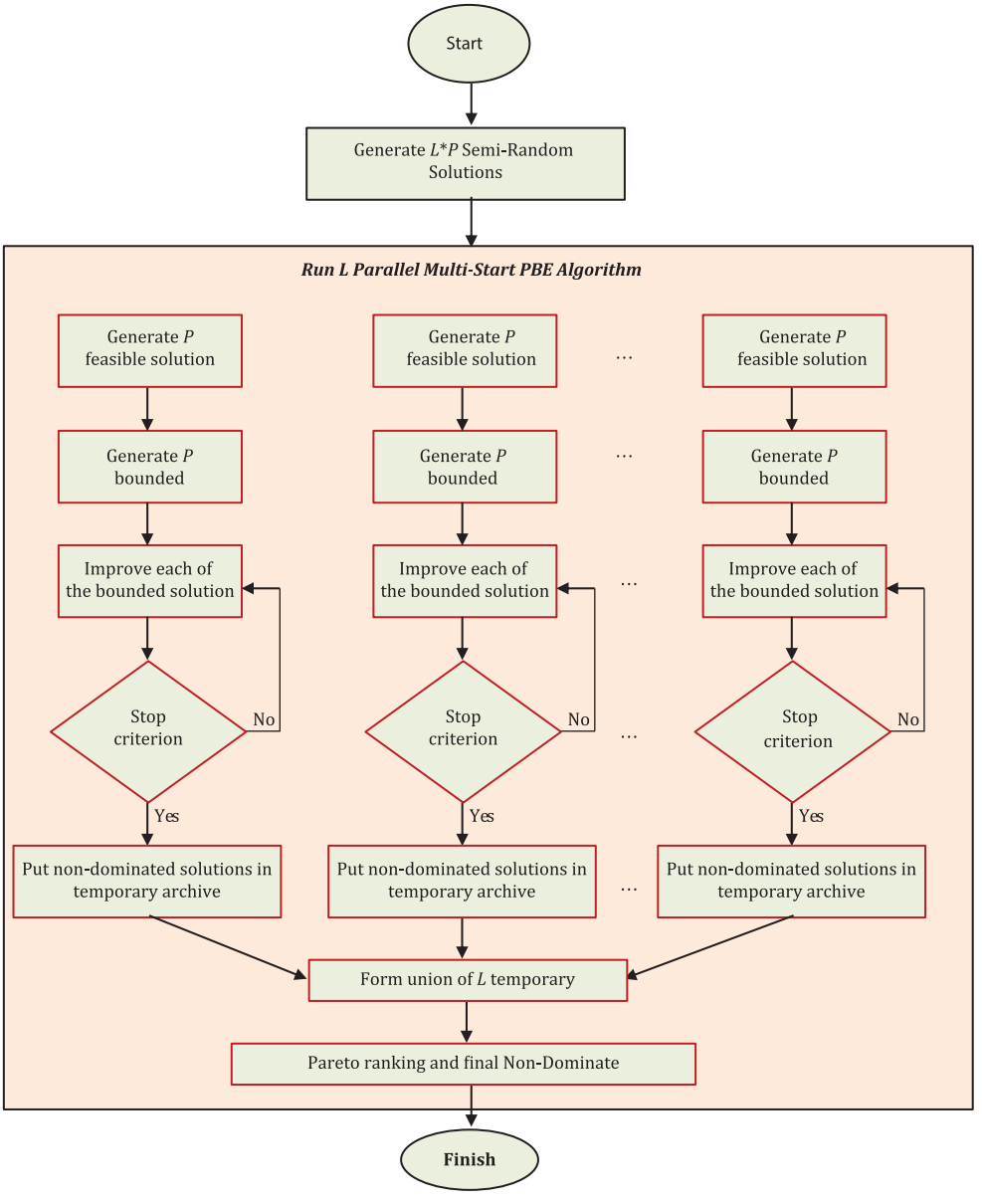


Figure 6. Parallel MSPBEA.

space are candidates for checking. Moreover, using a systematic approach, all bound solutions are not required to be checked. So the search procedure is very efficient.

Although the bound enumeration algorithm can achieve the real Pareto front of an MODM problem, checking the feasibility of a solution against different set of constraints, finding bound solutions, selecting non-dominated solutions among the feasible solutions, putting them in a temporary archive of non-dominated solutions in each step of the algorithm, and re-generating the true Pareto front of an MODM problem may be computationally expensive.

The multi-start property has been inserted in the proposed algorithm to utilize the capabilities of parallel computations in searching the feasible bound solutions of the TCQTP. In this regard, several local bound enumeration algorithms (i.e. L number, where L is a parameter of the algorithm) with different bound solutions are run in our proposed MSPBEA to efficiently search the solution space of the TCQTP. After several iterations of the proposed MSPBEA, whose number is a parameter of our proposed algorithm, the archives of the local PBEAs are compared to delete the dominated solutions. This property will lead to faster convergence to the Pareto front of TCQTP. Figure 6 presents the schematic view of the proposed multi-start PBEA for TCQTP.

5. Computational experiments

5.1 *Problem generation*

We systematically generated several random instances of the discrete generalized precedence TCQTP to measure the performance of the classical and EEC methods. We used a three-phase procedure in problem generation including a network design phase, a GPRs phase, and an activities' execution mode generation phase. For the first phase of the problem generation, we have used RanGen proposed by Demeulemeester *et al.* [12] to generate instances of project networks. Table 4 illustrates the characteristics of the generated random instances used in our computational experiments [2].

We should note that the order strength for all three classes is set to 1. The order strength is the number of precedence relations divided by the theoretical maximum number of precedence relations in the network.

We have generated instances in three classes, that is, small size, medium size, and large size. Ten random instances have generated in each class. The parameters of each class have been modelled as a uniform probability density function in order to make a balanced representation of the instances' properties. Each instance in each class is run 10 times and the results and metrics are discussed and calculated based on average results per class.

After the generation of the sample project networks, in order to assign the GPRs, time lags, and execution modes to the project activities, we have extended the procedure proposed by Tareghian and Taheri [46] which is illustrated in Figure 7. These phases were coded using VB 6.0.

It should be mentioned that different classes of RCPSPs and TCTPs have their own expertise benchmark instance libraries. For instance, the library of RCPSP/max is not the same as the library of RCPSP.

Table 4. Characteristics of the generated instances.

Class no.	Number of activities	Number of arcs	Complexity index[a]
1	10	45	5
2	50	1225	42
3	100	4950	87

[a]Measure of the closeness of a network to a series-parallel directed graph.

Generalized precedence relation generation:

1. *Categorize precedence relations as: 1=FF, 2=FS, 3=SF, 4=SS.*
2. *Randomly select the type of precedence relation for all of network's arcs by choosing an integer value in [1, 4].*

Time-lag assignment:

3. *Categorize time lags as: 1=Positive, 2=Negative.*
4. *Assign random sample for corresponding arc time lag in [0, 10].*

Execution mode generation:

5. *Choose number of modes (k) for activity i in [2, 15], randomly.*
6. *Generate time for mode r (for r=1, 2,..., k) in [10, 100], randomly.*
7. *Sort all execution modes of activity i based on their time in ascending order.*
8. *Generate execution cost for mode 1 in [1000, 5000], randomly.*
9. *Generate execution cost for other modes by* $c_i = c_{i-1} - s_i(t_i - t_{i-1})$, *where* s_i *randomly chosen in [1,5],* $s_1=1$, c_i, t_i *are the cost and time for mode i respectively.*
10. *Assign quality of mode 1 by 0.99.*
11. *Categorize quality into five levels in which each level can vary between 0.95…0.99, 0.90…0.99, 0.85…0.99, 0.80…0.99, 0.75…0.99.*
12. *Randomly select one quality level in [1, 5] to which activity i belongs to.*
13. *Assign random sample from the quality range of the corresponding quality level to mode r (for r=2,3,...,k).*

Figure 7. GPRs and execution mode generation algorithm.

5.2 *Software–hardware implementation*

The CEC and EEC methods were coded using LINGO 13.0 and VBA for MS-Excel 12.0 software. The DSAMOSO algorithm and MSPBE algorithm were coded using Matrix Laboratory (MATLAB) software. All codes were run on a Pentium IV PC with MS-Windows 7 Home edition, 2 GB of RAM, and 2.0 GHz Core 2 Due CPU.

5.3 *Parameter tuning*

We have tuned the parameters in all the methods based on a series of experimental analyses for small, medium, and large size classes. We have checked out the performance of the proposed methods (i.e. quality of the re-generated Pareto front and CPU time) in different settings in order to recognize the effective parameters experimentally. We have found that an archive size equal to 30 sets a proper trade-off between the performance of the proposed method and CPU time. The performance of all the algorithms and the effects of the parameters of the problem instances on CPU time are discussed in the next sub-sections.

In the current stage, we have obtained fitted parameters for all the solution methods (i.e. CEC, EEC, DSAMOPSO, and MSPBEA). Considering all the aforementioned trade-offs between the parameters of the proposed algorithms, the best-fitted values of the parameters presented in Table 5 have been categorized according to the problem benchmark instances (or classes).

The parameters in Table 5 have been determined after conducting DoE as follows.

Table 5. Fitted parameters for all the solution methods.

CEC Method		EEC method	
Problem Instance 1		Problem Instance 1	
Archive size	30	Archive size	30
Upper bound of cost	31,397	Upper bound of cost	31,397
Upper bound of quality	0.984	Upper bound of quality	0.984
Lower bound of cost	29,590	Lower bound of cost	29,590
Lower bound of quality	0.834	Lower bound of quality	0.834
Step size of cost	0.05	Step size of cost	0.05
Step size of quality	0.04	Step size of quality	0.04
Problem Instance 2		Problem Instance 2	
Archive Size	30	Archive size	30
Upper bound of cost	153,579	Upper bound of cost	153,579
Upper bound of quality	0.9786	Upper bound of quality	0.9786
Lower Bound of Cost	143,016	Lower bound of cost	143,016
Lower bound of quality	0.8222	Lower bound of quality	0.8222
Step size of cost	0.05	Step size of cost	0.05
Step size of quality	0.04	Step size of quality	0.04
Problem Instance 3		Problem Instance 3	
Archive size	30	Archive size	30
Upper bound of cost	314,310	Upper bound of cost	314,310
Upper Bound of Quality	0.9774	Upper Bound of Quality	0.9774
Lower bound of cost	293,269	Lower bound of cost	293,269
Lower bound of quality	0.8327	Lower bound of quality	0.8327
Step size of cost	0.05	Step size of cost	0.05
Step size of quality	0.04	Step size of quality	0.04
DSAMOPSO Algorithm		MSPBE Algorithm	
Problem Instance 1		Problem Instance 1	
Archive size	30	Archive size	30
Swarm size	100	No. of local algorithms	8
Iteration no.	300	No. of initial solutions	50
Cognitive factor	[0.04, 0.08]	Iteration no.	300
Social factor	[0.08, 0.04]		
α	1		
β	5		
Problem Instance 2		Problem Instance 2	
Archive size	30	Archive size	30
Swarm size	100	No. of local algorithms	8
Iteration no.	350	No. of initial solutions	100
Cognitive factor	[0.04, 0.08]	Iteration No.	350
Social factor	[0.08, 0.04]		
α	1		
β	5		
Problem Instance 3		Problem Instance 3	
Archive size	30	Archive size	30
Swarm size	100	No. of local algorithms	8
Iteration no.	400	No. of initial solutions	100
Cognitive factor	[0.04, 0.08]	Iteration No.	400
Social factor	[0.08, 0.04]		
α	1		
β	5		

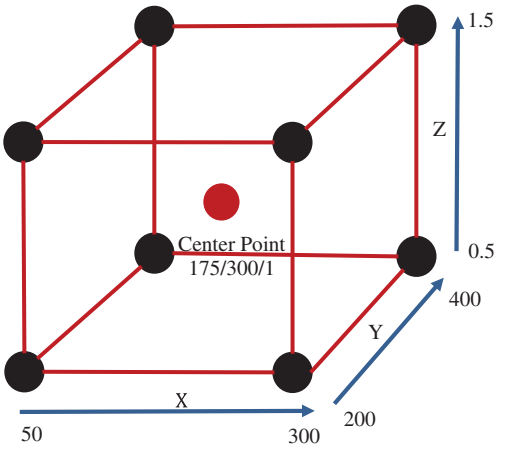


Figure 8. An illustration of a 3-factor full factorial DoE.

5.3.1 *Design of experiments (DoE)*

DoE is a stochastic framework for conducting representative experiments [17]. DoE attempts to minimize the amount of required experiments for an analysis, while maintaining high quality results. Experiments are considered to have input variables (factors) and output variables (responses). DoE provides a well-organized approach, combining experiments with extreme values and representative experiments, also called centre points, as represented in Figure 8. One of the main objectives in DoE is to optimize the factors by comparing and evaluating the quality of the responses. DoE has been successfully applied as a tool for the manual parameter tuning of particular computational optimization problems [31].

The initial stage of the process, experimental design, concerns the optimization of initial parameter values by applying an automated DoE to a finite training set of problem instances. Without loss of generality, let P be the set that compiles all instances of the problem at hand. $A_1, \ldots, A_m \in N$ are the initial parameter factors with their respective domains. Let $I_p = \{(a_1, \ldots, a_m) | a_j \in A_j, j \in m\}$ be the set of all possible initial parameter value combinations for P. DoE's factors for the factorial design are the solution procedure parameters $A_1, \ldots, A_m$, described in Table 5. That is, these parameters provide the input variables used in the factorial design performed via DoE.

Table 5 shows the final (best) points obtained for the experimental design, independent of P. Each benchmark instance has been assessed independently for the finding of near-optimal initial parameter values. For this reason, an iterative factorial design with repetitive experiments has been proposed. One iteration means conducting of a full-scale factorial design with extreme value and centre point simulations. Finding good extreme values for the parameters is a non-trivial task in itself. An investigation of the former achievements can reveal commonly applied settings that can work as an indicator for where minimal and maximal bounds could be settled. Therefore, the results from a former iteration have been used for screening the most promising areas of the search space.

5.3.2 *Initial parameters for the CEC and EEC methods*

For both the CEC and EEC methods, the upper bound of cost function has been calculated based on the maximization of the cost objective function (i.e. Equation (2)) subject to constraints (4)–(13). The lower bound of cost function has been calculated based on the minimization of the

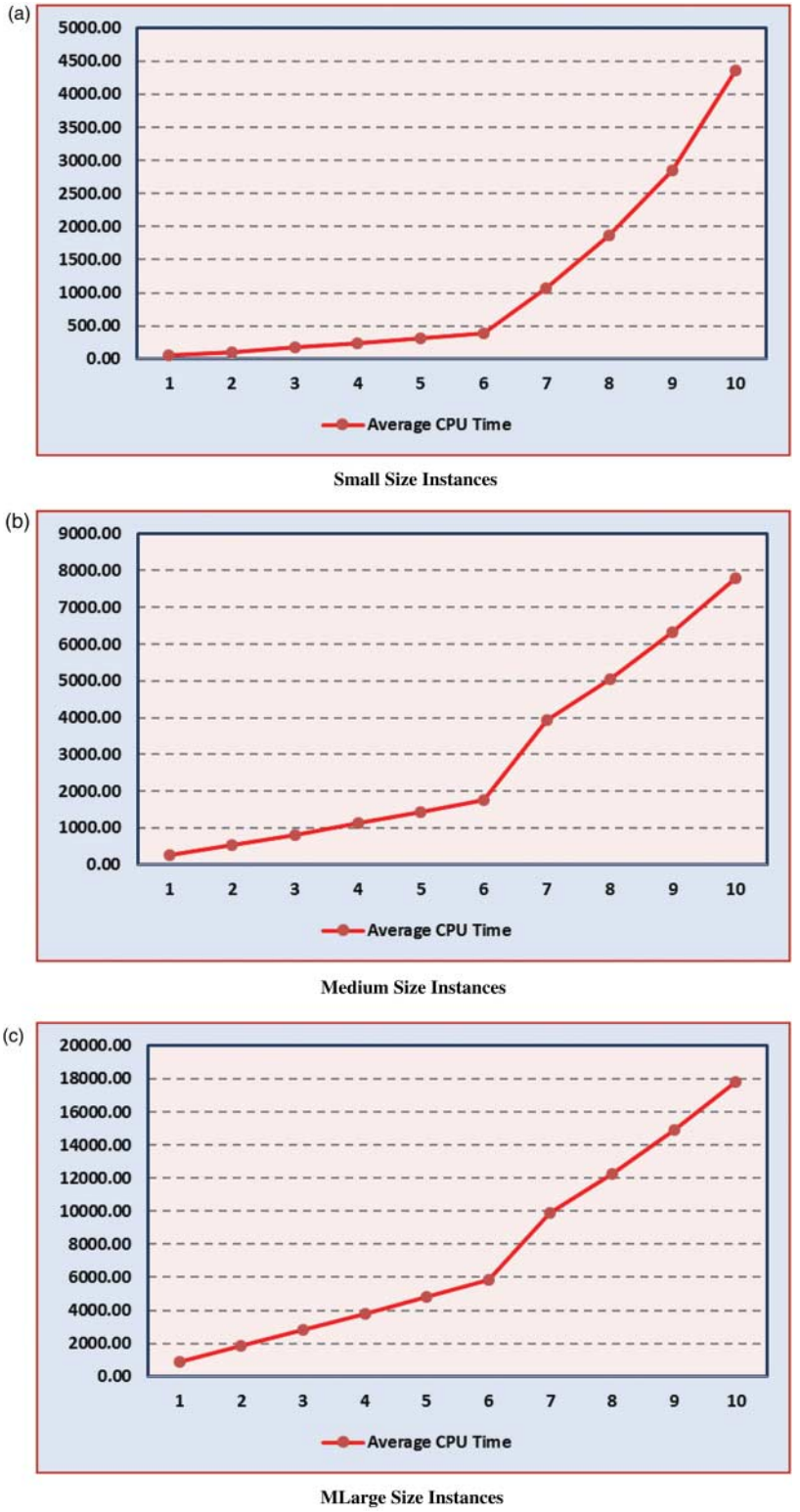


Figure 9. Archive size versus average CPU time.

Table 6. Average CPU time versus archive size.

	Average CPU time		
Archive size	Small size	Medium size	Large size
5	52.50	257.50	912.50
10	110.00	530.00	1850.00
15	172.50	817.50	2812.50
20	240.00	1120.00	3800.00
25	312.50	1437.50	4812.50
30	390.00	1770.00	5850.00
35	1074.72	3924.16	9923.60
40	1863.38	6390.13	14,516.88
45	2854.44	9463.31	18,424.16
50	4365.62	14,096.86	28,328.09

cost objective function (i.e. Equation (2)) subject to constraints (4)–(13). The lower bound of quality function has been calculated based on the minimization of the quality objective function (i.e. Equation (3)) subject to constraints (4)–(13). The upper bound of quality function has been calculated based on the maximization of the quality objective function (i.e. Equation (3)) subject to constraints (4)–(13).

The step size of the cost and quality objective functions have been set equal to 0.05 and 0.04, respectively, within the ranges of the cost and quality objective functions. As the step size of the cost and quality objectives are set equal to a smaller value, the density of the Pareto front may increase but the CPU time of the algorithm could increase dramatically. Moreover, a very small value for the step size of the cost and quality functions, that is, lower than 0.01, generated several similar non-dominated solutions while the CPU time was increased inefficiently. On the other hand, a step size larger than 0.1 generated a very low number of non-dominated solutions and this made the estimated Pareto front sparse. We have tested the step sizes 0.01, 0.02, 0.03. 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, and 0.1 and have found that 0.05 and 0.04 are the most suitable values for the cost and quality objectives, respectively.

The best archive size has been set equal to 30 based on a trade-off between the number of generated non-dominated solutions and the CPU time. We have tested 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50 archive sizes. Figure 9(a)–(c) presents a 2-D plot of the number of non-dominated solutions and the CPU time for small size, medium size, and large size benchmark instances, respectively.

As shown in Figure 9(a)–(c), the CPU time had a meaningful spike when the archive size was bigger than 30. Hence, we choose an archive size equal to 30 to maintain a proper trade-off between the CPU time and the number of non-dominated solutions in the Pareto front. The numerical results are also presented in Table 6.

5.3.3 *Initial parameters for the DSAMOPSO and MSPBE algorithms*

We have set the archive size for the DSAMOPSO and MSPBE algorithms equal to 30 in order to create a fair environment for comparing their performance with that of the CEC and the EEC methods.

The number of local algorithms in MSPBE has a direct effect on the CPU time and the number of non-dominated solutions obtained. We have tested 2, 4, 8, 16, and 32 for the number of local algorithms. When the number of parallel implementations of the local algorithms was set to less than 4, the CPU time was suitable but the quality and the number of non-dominated solutions were not desirable. When it was set greater than 16, the CPU time was increased dramatically,

Table 7. Comparison metrics.

Type	Metric	Description	Definition
Convergence metrics	NNS	The number of non-dominated solutions that each procedure can find	
	ER	Error rate: measures the non-convergence of the algorithms towards real Pareto front	$\mathrm{ER} = \frac{\sum_{i=1}^{n} e_i}{n}$, $e_i = \begin{cases} 0 & \text{Solution } i \text{ belongs to Pareto front} \\ 1 & \text{otherwise} \end{cases}$
	GD	Generational distance: Calculates the distance between RS and the solution set	$\mathrm{GD} = \frac{\sum_{i=1}^{n} d_i}{n}$, $d_i = \min_{p \in \mathrm{PF}} \left\{ \sqrt{\sum_{k=1}^{m} (z_k^i - z_k^p)^2} \right\}$; where m is the number of objective values and PF is the reserved for Pareto Front
Distribution metrics	SM	Spacing metric: measures the uniformity of the spread of the points of the solution set	$\bar{d} = \frac{\sum_{i=1}^{n} d_i}{n}$, $\mathrm{SP} = \sqrt{\frac{\sum_{i=1}^{n} (\bar{d} - d_i)^2}{n-1}}$
	DM	Diversification metric: measures the spread of the solution set	$\mathrm{DM} = \left[\sum_{i=1}^{n} \max(x_i - y_i) \right]^{1/2}$; where $x_i - y_i$ is the Euclidean distance between two non-dominated solutions x_i and y_i

but the increment in the number of non-dominated solutions was considerable. We found that the best trade-off is achieved when the number of local algorithms is set equal to 8.

The swarm size and the number of initial solutions in the DSAMOPSO and MSPBE algorithms has been set equal to 100. We tested 50, 100, 150, and 200 for the swarm size and the number of initial solutions. We found out that a size of 100 provides a proper trade-off between the number of non-dominated solutions and the CPU time. The analysis performed is similar to those for determining the archive size.

The iteration number is another important parameter which influences the number of non-dominated solutions and the CPU time. An iteration number lower than 300 exhibited a reasonable CPU time but the number of non-dominated solutions could not fill out the archive size. Iteration numbers greater than 400 generated suitable non-dominated solutions but the CPU time was increased dramatically. We have tested 5 values for the iteration numbers, including 250, 300, 350, 400, and 450. We found that the best value for the iteration number is 300, 350, and 400 for the small, medium, and large size benchmark instances, respectively.

The cognitive factor, social factor, alpha, and beta parameters were benchmarked based on the successful implementations described in the literature [23,49].

5.4 *Comparison metrics*

We have used two sets of accuracy and diversity metrics to provide a basis for evaluating the relative performance of the CEC, EEC, DSAMOPSO, and MSPBE methods. These metrics have been proposed by Yu and Gen [51] to extensively compare multi-objective optimization algorithms. The metrics and their definitions have been summarized in Table 7.

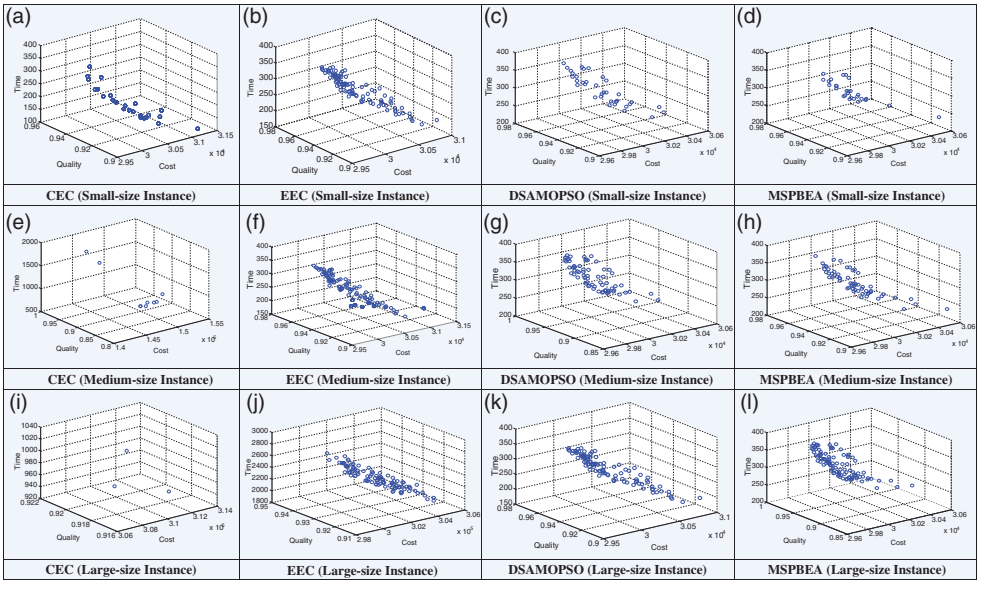


Figure 10. Generated non-dominated solutions for CEC, EEC, DSAMOPS, and MSPBEA methods.

5.5 *Results*

The CEC and EEC were coded in LINGO software. DSAMOPSO and MSPBEA were coded in MATLAB software. All generated instances were solved using these codes. Figure 10 represents the non-dominated solutions generated by each method for all sizes of benchmark instances.

As shown in Figure 10, in all benchmark instances, the generated Pareto fronts for the CEC method are sparse while these are rather dense for the EEC, DSAMOPSO, and MSPBEA methods. The situation is even worse for large-scale instances. As shown in this figure, the CEC method generates a limited number of distinct non-dominated solutions over the real Pareto front while the other methods (i.e. EEC, DSAMOPSO, and MSPBEA) are capable of generating denser sets of non-dominated solutions for medium and large size instances. Table 8 presents a comparative analysis of the different objective functions obtained from 30 non-dominated solutions for the CEC, EEC, DSAMOPSO, and MSPBEA methods for all categories of the benchmark instances.

The average, standard deviation, minimum, and maximum values of time, cost, and quality objective functions, for all methods, including CEC, EEC, DSAMOPSO, and MSPBEA, and all class sizes are presented in Table 8. These values represent the relative performance of all four methods. Note how time exhibits the highest variability in the average values obtained as the size of the benchmark instance increases, a result particularly relevant when considering the interactions of different projects within a given chain.

As the real Pareto front of the benchmark instances is not known in advance, a reference set (RS) was defined for each benchmark instance. A RS is the set of best known solutions for the benchmark instances obtained throughout the 10 different runs of all methods. Figure 11 presents the RSs for the three categories of the benchmark instances.

Although the relative dominance of the proposed methods (i.e. EEC, DSAMOPSO, and MSPBEA) was illustrated in Table 8 and Figure 10, we have organized a more detailed analysis in order to represent the weaknesses and strengths exhibited by all the methods through the runs. We have used several measures to test the accuracy and diversity of the non-dominated solutions generated by the different methods.

Table 8. Objective function results for all methods.

Objective method	Statistical measure	Time				Cost				Quality			
		CEC	EEC	DSAMOPSO	MSPBEA	CEC	EEC	DSAMOPSO	MSPBEA	CEC	EEC	DSAMOPSO	MSPBEA
Small size	Average	221.00	221.63	210.83	215.73	30,525.53	30,530.93	30,410.77	30,481.67	92.90	92.73	93.90	92.70
	Std. Dev.	16.57	17.07	10.74	16.52	259.38	325.66	263.50	300.15	2.09	1.82	1.88	1.76
	Min	191.00	193.00	193.00	192.00	30,062.00	30,022.00	30,038.00	30,001.00	90.00	90.00	90.00	90.00
	Max	246.00	249.00	234.00	248.00	30,970.00	30,996.00	30,967.00	30,935.00	96.00	96.00	96.00	96.00
Medium size	Average	939.87	481.65	660.73	563.92	149,307.50	150,622.27	149,586.73	149,968.93	92.93	93.03	92.67	92.73
	Std. dev.	81.90	172.82	163.84	49.14	2427.19	2279.00	2058.86	2156.17	2.07	1.50	1.83	1.74
	Min	802.00	186.44	372.00	481.20	146,054.00	146,313.00	146,021.00	146,158.00	90.00	91.00	90.00	90.00
	Max	1092.00	822.74	973.50	655.20	153,907.00	153,980.00	153,881.00	153,790.00	96.00	95.00	96.00	96.00
Large size	Average	1955.23	464.49	693.77	1173.14	300,082.30	299,926.00	299,825.87	300,561.17	70.25	70.14	70.28	70.12
	Std. dev.	90.26	166.41	156.95	54.16	2298.49	2147.69	2344.42	2064.20	2.25	1.53	1.89	1.62
	Min	1813.00	190.30	412.50	1087.80	296,183.00	296,309.00	296,257.00	297,088.00	90.00	90.00	90.00	90.00
	Max	2097.00	800.88	975.00	1258.20	303,632.00	303,958.00	303,833.00	303,668.00	96.00	96.00	96.00	96.00

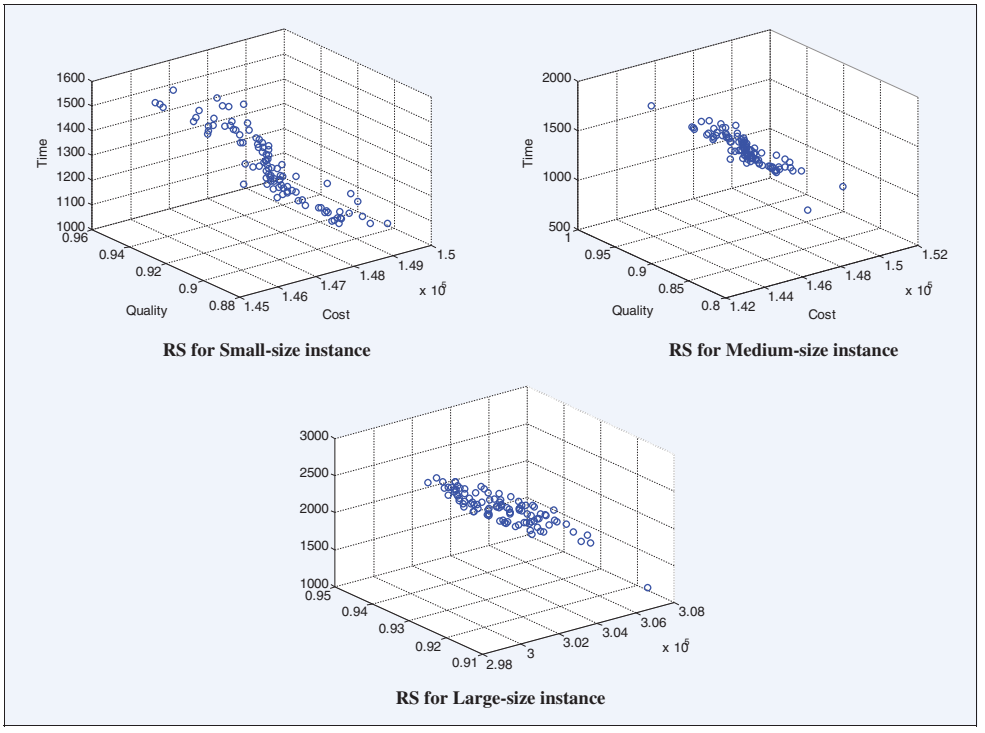


Figure 11. RSs for different benchmark instances.

5.5.1 *Result of comparison metrics*

Comparison metrics were calculated using the RSs for all benchmark instances over 10 different runs of all the methods. Tables 9–11 are presented to demonstrate the accuracy and diversity metrics obtained for small, medium, and large size benchmark instances.

Table 9 presents the result of applying accuracy and diversity metrics to small size instances. It can be concluded from Table 9, that DSAMOPSO has the highest NNS on average for all runs while the MSPBEA has the lowest. In other words, the DSAMOPSO method generates relatively more non-dominated solutions when compared to the other methods. The NNS metric measures the number of non-dominated solutions generated by a given method. It should be noted that all the solutions generated by a method are non-dominated when the method is implemented through multiple runs. However, a non-dominated solution obtained by a method may be dominated by the solutions generated by other methods. Therefore, when the NNS metric is computed, the performance of a given method is compared with all other methods using the RS.

Consider now the error rate metric. The MSPBEA method has the biggest ER on average among the other methods while the DSAMOPSO has the lowest ER on average. The ER metric measures the non-convergence of a given method towards the Pareto front. In other words, the ER is a measure of performance of a given method in generating Pareto front. For a given method, the ER is calculated as the number of solutions that do not belong to the Pareto front (or RS) divided by the number of solutions composing the Pareto front (or RS). For a given method, the ER will be equal to zero if all generated solutions are on the Pareto front (or RS).

The last convergence metric is GD, in which the distance between the RS and the Pareto front re-generated by a given method is calculated. The GD is calculated as the minimum Euclidean distance between a re-generated solution and all solutions in the Pareto front, defined within a

Table 9. Computational results for the small instance.

	Accuracy metrics						Diversity metrics				Run time	
	NNS		ER		GD		SM		DM		CPU time (s)	
Run	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC
1	42	42	0.73	0.78	178	203	64	98	205	202	13	75
2	43	35	0.74	0.7	231	265	56	86	200	200	10	57
3	30	37	0.88	0.48	147	163	32	66	205	199	14	63
4	30	42	0.13	0.48	200	298	49	73	210	205	11	60
5	37	41	0.66	0.07	206	198	58	82	204	203	15	58
6	43	42	0.88	0.69	154	287	43	90	201	201	10	62
7	40	31	0.54	0.68	161	232	48	59	200	201	10	55
8	39	36	0.38	0.49	160	160	50	46	207	203	14	52
9	36	25	0.66	0.64	162	252	71	70	210	201	13	70
10	30	25	0.65	0.36	130	255	73	45	206	201	15	59
Ave.	37	35.6	0.625	0.537	172.9	231.3	54.4	71.5	204.8	201.6	12.5	61.1
Std. dev.	5.35	6.67	0.23	0.21	30.78	48.69	12.69	17.98	3.68	1.71	2.07	6.90
	Accuracy metrics						Diversity metrics				Run time	
	NNS		ER		GD		SM		DM		CPU time (S)	
Run	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA
1	44	25	0.17	0.47	243	190	92	57	208	204	16	31
2	33	30	0.81	0.89	157	217	89	40	206	206	25	38
3	46	33	0.48	0.86	150	126	91	44	205	208	20	43
4	38	27	0.67	0.92	174	114	100	55	207	207	16	38
5	32	34	0.18	0.9	115	152	99	72	205	208	21	34
6	40	40	0.81	0.87	128	276	68	43	207	203	22	35
7	33	38	0.15	0.13	243	271	93	44	208	203	25	39
8	46	26	0.48	0.84	91	289	61	66	210	204	21	37
9	45	29	0.12	0.52	158	164	69	40	208	208	17	42
10	34	28	0.65	0.61	200	106	51	53	210	208	17	44
Ave.	39.1	31	0.452	0.701	165.9	190.5	81.3	51.4	207.4	205.9	20	38.1
Std. dev.	5.84	5.10	0.28	0.26	50.72	69.55	17.40	11.18	1.78	2.18	3.43	4.12

Table 10. Computational results for the medium instance.

	Accuracy metrics						Diversity metrics				Run time	
	NNS		ER		GD		SM		DM		CPU time (s)	
Run	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC
1	39	31	0.67	0.36	424	475	112	131	608	608	42	157
2	38	37	0.22	0.47	539	830	202	153	621	612	30	222
3	34	34	0.95	0.06	482	478	137	230	619	598	37	152
4	41	31	0.21	0.1	428	476	192	142	605	598	34	223
5	34	36	0.96	0.83	415	831	109	188	627	597	37	219
6	44	28	0.34	0.05	538	861	146	165	603	601	38	196
7	31	39	0.76	0.22	476	820	118	203	612	609	41	186
8	42	38	0.13	0.69	344	534	240	249	630	614	30	205
9	31	32	0.79	0.15	491	526	105	164	602	607	30	214
10	43	38	0.04	0.14	412	821	151	215	603	612	31	192
Ave.	37.7	34.4	0.507	0.307	454.9	665.2	151.2	184	613	605.6	35	196.6
Std. dev.	4.90	3.75	0.35	0.27	61.35	177.90	45.84	39.40	10.52	6.52	4.64	25.62
	Accuracy metrics						Diversity metrics				Run time	
	NNS		ER		GD		SM		DM		CPU Time (S)	
Run	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA
1	37	33	0.78	0.82	453	548	150	165	626	622	59	101
2	44	25	0.91	0.2	361	544	187	122	623	615	59	117
3	38	40	0.25	0	499	563	221	130	628	613	60	100
4	36	26	0.36	0.11	255	434	275	232	626	620	58	117
5	37	25	0.78	0.18	325	308	246	231	615	620	58	106
6	43	37	0.04	0.05	443	392	275	234	629	615	72	125
7	38	27	0.14	0.69	626	332	259	222	623	624	52	129
8	33	38	0.87	0.11	296	526	203	160	621	621	74	120
9	47	36	0.66	0.62	718	548	171	210	615	616	51	134
10	31	38	0.08	0.24	575	476	291	179	630	609	49	100
Ave.	38.4	32.5	0.487	0.302	455.1	467.1	227.8	188.5	623.6	617.5	59.2	114.9
Std. dev.	4.95	6.10	0.35	0.29	151.26	95.48	48.72	43.01	5.34	4.65	8.23	12.55

Table 11. Computational results for the large instance.

	Accuracy metrics						Diversity metrics				Run time	
	NNS		ER		GD		SM		DM		CPU time (s)	
Run	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC	EEC	CEC
1	35	27	0.54	0.82	1066	1486	237	546	1241	1203	71	435
2	40	33	0.95	0.93	729	930	223	547	1214	1198	69	419
3	34	34	0.16	0.52	721	875	338	486	1253	1209	67	332
4	34	42	0.79	0.51	1029	720	195	546	1219	1198	76	385
5	36	42	0.18	0.35	843	1781	350	253	1224	1207	88	419
6	35	26	0.85	0.13	949	982	420	519	1210	1188	87	402
7	43	34	0.27	0.91	679	1582	276	324	1205	1218	79	354
8	39	27	0.91	0.31	1026	1512	323	312	1252	1189	60	364
9	36	42	0.97	0.19	921	731	229	580	1200	1199	89	375
10	34	29	0.94	0.01	1009	1637	344	254	1237	1208	76	331
Ave.	36.6	33.6	0.656	0.468	897.2	1223.6	293.5	436.7	1225.5	1201.7	76.2	381.6
Std. dev.	3.06	6.48	0.34	0.33	144.56	411.56	72.21	133.82	19.20	9.26	9.74	36.84
	Accuracy metrics						Diversity metrics				Run time	
	NNS		ER		GD		SM		DM		CPU time (S)	
Run	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA	MOPSO	MSPBEA
1	32	41	0.42	0.94	1377	1043	357	422	1256	1241	147	244
2	38	41	0.36	0.44	499	959	542	312	1256	1219	145	249
3	45	25	0.85	0.52	992	947	554	436	1238	1234	125	196
4	45	32	0.36	0.28	911	726	490	419	1241	1242	143	237
5	32	39	0.98	0.88	767	695	451	323	1258	1233	134	190
6	36	32	0.18	0.97	729	992	335	319	1252	1229	145	207
7	31	39	0.84	0.83	1467	1134	584	299	1245	1242	99	258
8	33	27	0.62	0.03	1486	630	384	328	1244	1237	132	232
9	41	31	0.74	0.63	1360	823	331	458	1253	1246	135	197
10	42	27	0.9	0.67	630	681	429	278	1260	1236	110	199
Ave.	37.5	33.4	0.625	0.619	1021.8	863	445.7	359.4	1250.3	1235.9	131.5	220.9
Std. dev.	5.48	6.15	0.28	0.31	371.91	174.73	93.98	66.27	7.70	7.81	16.04	25.61

m-dimensional objective function space, divided by number of solutions in the Pareto front (or RS). The DSAMOPSO method has the lowest GD while the CEC method has the largest GD value on average. This means that the Pareto front generated by the DSAMOPSO method has relatively the lowest distance, that is, is the one located closest, to the solutions in the Pareto front (or RS).

In addition to the accuracy of the generated Pareto front, its diversity is also very important. We have used SM and DM in order to measure the diversity of re-generated Pareto front. The SM metric measures the uniformity of the spread of the points composing the solution set. This metric measures the uniformity of the solutions generated by a given method in comparison with the solutions on the Pareto front. A higher value of the SM means that the associated method generates more uniform solutions on the Pareto front. The DSAMOPSO method has the highest SM values while the MSPBEA method has the lowest ones, on average. In other words, the solutions generated by the DSAMOPSO method have been distributed uniformly on the Pareto front and the DSAMOPSO method has been successful in re-generating all parts of the Pareto front (or RS) uniformly.

The DM metric measures the spread of the solution set. In other words, this metric measures the maximum Euclidean distance between the generated solutions by a given method. No comparison with the real Pareto front or RS is done when calculating DM. A big value for DM means that the method is successful in generating a front with suitable spread. The biggest DM measure is for the DSAMOPSO method while the lowest DM measure is for the CEC method. Therefore, the solutions generated by the DSAMOPSO method are spread throughout the entire re-generated front.

In summary, the DSAMOPSO method provides the best representation of the RS among all tested methods, covering uniformly a larger area of the Pareto front.

Finally, in order to complement the analysis based on the comparison metrics, we consider the CPU time required by each method. In this regard, we can observe that the lowest CPU time is required by the EEC method while the largest CPU time is that of the CEC method.

Tables 10 and 11 present the results of the accuracy and diversity metrics for medium and large size classes, respectively. It should be noted that the interpretation of the results for medium and large size benchmark instances is similar to that of the small ones. Hence, the descriptions will not be repeated for sake of brevity.

5.5.2 *Comparative analysis*

Although the discussion of the contents of Tables 9–11 sheds some light on the performance of all the algorithms, some additional and more general comparisons may be required to get a better idea of the overall performance of each method. Table 12 presents the position achieved (from first to last) by all the algorithms for all the benchmark instances.

Table 12 presents the ranks of all the methods (i.e. CEC, EEC, DSAMOPSO, and MSPBEA) for all class sizes. That is, it summarizes the overall performance of all the methods for all sizes of benchmark instances.

We must highlight that the proposed DSAMOPSO algorithm got the first rank for five metrics (i.e. NNS, ER, GD, SM, and DM) when considering the small size class. It also got ranked second for CPU time. On the other hand, both the CEC method and the MSPBEA algorithm got ranked in the fourth position for three different metrics.

When considering the medium size class, the DSAMOPSO algorithm presents also an excellent performance compared to the other methods. DSAMOPSO got the first rank for three metrics in the medium size instance (i.e. NNS, SM, and DM). Once again the first ranks for NNS, SM, and DM are assigned to the DSAMOPSO algorithm in the large size class.

Table 12. Comparative analysis of methods for all instances: rank positions based on the averages obtained from all runs.

Methods	Rank NNS	Rank ER	Rank GD	Rank SM	Rank DM	Rank CPU time (s)
Small size instance						
CEC	3	2	4	2	4	4
EEC	2	3	2	3	3	1
DSAMOPSO	1	1	1	1	1	2
MSPBEA	4	4	3	4	2	3
Medium size instance						
CEC	3	2	4	3	4	4
EEC	2	4	1	4	3	1
DSAMOPSO	1	3	2	1	1	2
MSPBEA	4	1	3	2	2	3
Large size instance						
CEC	3	1	4	2	4	4
EEC	2	4	2	4	3	1
DSAMOPSO	1	3	3	1	1	2
MSPBEA	4	2	1	3	2	3

Table 13. Average ranking for all instance sizes.

Methods	NNS	ER	GD	SM	DM	CPU Time (S)	Average
CEC	3	1.67	4	2.33	4	4	3.166667
EEC	2	3.67	1.67	3.67	3	1	2.501667
DSAMOPSO	1	2.33	2	1	1	2	1.555
MSPBEA	4	2.33	2.33	3	2	3	2.776667

The ranking values of Table 12 illustrate the relative dominance of the proposed DSAMOPSO algorithm for all class sizes. We have however provided a more general ranking in Table 13 in order to rank all methods based on their overall performances in all benchmark instances using all metrics.

We have calculated the average ranking of all the methods considering each metric in all class sizes and presented them in Table 13. The DSAMOPSO got the first rank for most of the metrics in all class sizes. Moreover, we have calculated the average value of the average ranks for each method and presented it in the last column of Table 13. The best method when averaging through all runs, for all instance sizes, and according to all metrics (i.e. NNS, ER, GD, SM, DM, and CPU time) is DSAMOPSO with an overall average rank equal to 1.5. The second overall place is for the EEC method and MSPBEA and CEC got the third and fourth places, respectively.

As mentioned in the introduction, the TCQTPs are NP-Hard problems with a large solution space (7–8, 14). Therefore, it is not possible to develop a polynomial time order algorithm for medium and large size instances of the NP-Hard TCQTPs. Although the exact algorithms (i.e. CEC and EEC methods) showed a suitable performance for small size instances, they leave the competition in favour of heuristic methods (i.e. DSAMOPSO and MSPBEA) as the size of the benchmark instances increases. In this latter case, the exact methods cannot fill out the archive size and the number of non-dominated solutions generated decreases dramatically. These results were somehow predictable since the CEC and EEC methods cannot investigate the increased solution spaces of large size instances efficiently. In this regard, based on the suitable performance of heuristic methods in small size instances, decision-makers should be satisfied with the solutions generated by the heuristic methods.

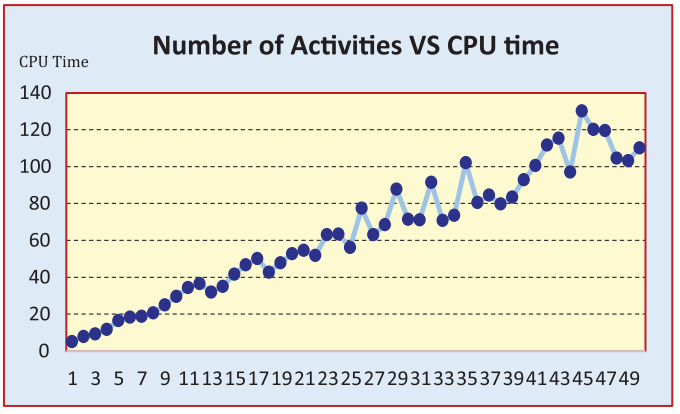


Figure 12. Number of activities versus CPU time.

Table 14. Pearson correlation test results between CPU time and number of activities.

	Pearson correlation	CPU time	No. of activities
CPU time	Pearson correlation	1	.977**
	Sig. (2-tailed)		.000
	N	50	50
No. of activities	Pearson correlation	.977**	1
	Sig. (2-tailed)	.000	
	N	50	50

**Correlation is significant at the 0.01 level (2-tailed).

5.5.3 *Effect of the number of parameters on CPU time*

In order to highlight the effect that the parameters of the benchmark instances have on CPU time, two tests were carried out using DSAMOPSO, which has been shown to be the best algorithm compared to the others.

First test: Number of activities versus CPU time. In the first test, the relation between the number of activities in the project network and the CPU time of the proposed DSAMOPSO is measured. The number of activities was selected from the interval [10, 500] randomly. All other parameters were assumed to be constant and under control in all 50 generated instances, which are displayed in increasing size order on the horizontal axis of Figure 12. The order strength was set equal to 1 and the complexity index was set to the maximum value for all instances. In this way, the only effective parameter affecting the CPU time was the number of activities. Note, however, that some uncontrollable parameters such as the performance of hardware during test runs always exists. As is clear from Figure 12, a semi-linear relationship can be observed between the number of activities in the project and the CPU time of the proposed DSAMOPSO algorithm. The random noises can be interpreted as effects of the uncontrollable parameters in this test.

We have performed the Pearson correlation test between both variables in Statistical Package for the Social Sciences software in order to test if there is a linear relation between the number of activities in the project and the CPU time of the proposed DSAMOPSO algorithm. The results of the Pearson correlation test are presented in Table 14.

It can be concluded from Table 14 that there is a high positive linear correlation equal to 0.977 between the number of activities in the project and the CPU time of the proposed DSAMOPSO algorithm. This high linear correlation validates the hypothesis that the CPU

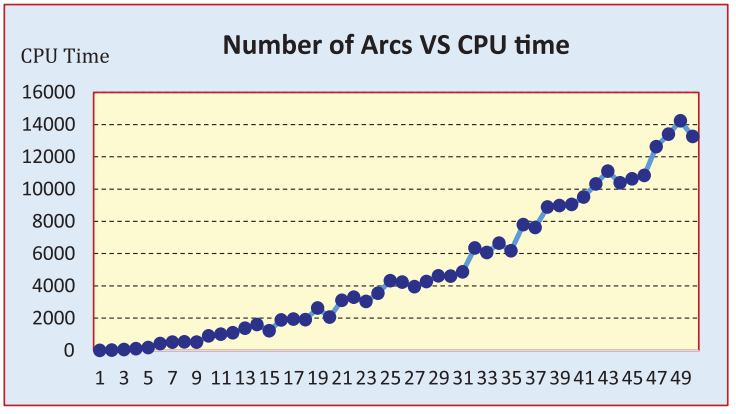


Figure 13. Number of arcs versus CPU time.

Table 15. Pearson correlation test results between CPU time and number of arcs.

	Pearson correlation	CPU time	No. of arcs
CPU time	Pearson correlation	1	.231**
	Sig. (2-tailed)		.000
	N	50	50
No. of arcs	Pearson correlation	.231**	1
	Sig. (2-tailed)	.000	
	N	50	50

**Correlation is significant at the 0.01 level (2-tailed).

time of the algorithm increases linearly with the number of activities in the project. This is an advantageous property for the proposed DSAMOPSO algorithm.

Second test: number of arcs versus CPU time. In the second test, the relation between the number of arcs in a project network and the CPU time of the proposed DSAMOPSO was analysed. In this test the number of activities of all instances were fixed and set equal to 100. Fifty distinctive networks were formed and the number of arcs in the networks were selected from the interval [2475, 4590]. These networks are displayed in increasing order, based on the number of arcs, on the horizontal axis of Figure 13. It should be emphasized that a fully connected network with 100 nodes can have at most 4590 arcs. The complexity index was set to the maximum for all instances. Again, we wanted to find out the effect of the number of arcs in a project network on the CPU time of the proposed DSAMOPSO. As is clear from Figure 13, the relationship between the number of arcs in the project and the CPU time of the proposed DSAMOPSO algorithm is nonlinear. The CPU time is increased substantially as the number of arcs in the project increases. The random noises can also be observed in this test.

The Pearson correlation test was also performed between the number of arcs in the project and the CPU time of the proposed DSAMOPSO algorithm. The results of the Pearson correlation test are presented in Table 15.

It can be concluded from Table 15 that there is a weak positive linear correlation equal to 0.23 between the number of arcs in the project and the CPU time of the proposed DSAMOPSO algorithm. This weak linear correlation validates the hypothesis that the CPU time of the algorithm is weakly correlated with the number of arcs in the project. Therefore, the CPU time of the proposed DSAMOPSO is not a linear function of the number of arcs in the projects and this parameter imposes more computational effort to the algorithm.

6. Conclusion and future research directions

The activities in real-world project scheduling problems can be accomplished using multi-mode execution processes with different time, cost, and quality values. In addition, project scheduling problems are subject to different precedence relations among their activities. Finally, project scheduling problems are generally constrained by resources in discrete units. In one type of project scheduling problem, called trade-off problems, all resources are considered as a cost to the project. In this study, we proposed a new mixed-integer mathematical formulation for solving the discrete GPRs multi-objective multi-mode TCQTPs. In discrete generalized precedence TCQTPs, the decision-makers require a set of non-dominated solutions instead of a single solution since there is no articulation of their preferences in these problems.

Four multi-objective solution procedures, including the CEC method, the EEC method, DSAMOPSO, and MSPBEA, were used to solve the proposed model. We simulated sets of random problems to measure and compare the efficiency and applicability of the proposed methods. The Pareto fronts generated by all these methods were compared with respect to a RS (i.e. the best known Pareto front generated through the runs on the instances of the discrete generalized precedence TCQTPs) based on two sets of diversification and convergence metrics (i.e. NNS, ER, DM, GD, and SM) together with the CPU time required for multi-objective optimization. A quantitative analysis was performed to summarize the performance of all the methods using their ranks in all the proposed metrics. The DSAMOPSO method performed relatively better than the other methods. Moreover, sensitivity analysis was performed to illustrate the effects of the parameters of the project on CPU time using the best method, that is, the DSAMOPSO algorithm. The effect of the number of activities and arcs on the CPU time of the algorithm was measured using two sets of controlled tests.

The project and activities parameters, such as cost, time, and quality, usually change through the planning horizon. These changes can affect the design of the trade-off problem. Some form of uncertainty modelling using techniques, such as interval data, fuzzy sets or even probabilistic programming, can also be the subjects of future research.

Acknowledgements

The authors would like to thank the anonymous reviewers and the editor-in-chief for insightful comments and suggestions. This research has been accomplished as a research plan entitled ‘Solving Multi-Mode Time–Cost–Quality Trade-Off Problem under Generalized Precedence Relations using an Efficient Multi-Objective Method’. This research is supported by South-Tehran Branch, Islamic Azad University, Tehran, Iran.

Note

1. Project Scheduling Problem Library: PSPLIB, http://www.om-db.wi.tum.de/psplib/main.html.

References

[1] A. Afshar, A. Kaveh, and O. Shoghli, *Multi-objective optimization of time–cost–quality using multi-colony ant algorithm*, Asian J. Civil Eng. (Build. Housing) 8 (2007), pp. 113–124.
[2] M. Amiri, A.R. Abtahi, and K. Khalili-Damghani, *Solving a generalised precedence multi-objective multi-mode time–cost–quality trade-off project scheduling problem using a modified NSGA-II algorithm*, Int. J. Serv. Oper. Manage. 14 (2013), pp. 355–372.
[3] A.J. Babu and N. Suresh, *Project management with time, cost and quality considerations*, Eur. J. Oper. Res. 88 (1996), pp. 320–327.
[4] L. Bianco and M. Caramia, *A new lower bound for the resource-constrained project scheduling problem with generalized precedence relations*, Comput. Oper. Res. 38 (2011), pp. 14–20.

[5] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, *Resource-constrained project scheduling: Notation, classification, models, and methods*, Eur. J. Oper. Res. 112 (1999), pp. 3–41.
[6] P. Brucker and S. Knust, *Lower bounds for resource-constrained project scheduling problems*, Eur. J. Oper. Res. 149 (2003), pp. 302–313.
[7] P. De, E.J. Dunne, J.B. Ghosh, and C.E. Wells, *The discrete time–cost tradeoff problem revisited*, Eur. J. Oper. Res. 81 (1995), pp. 225–238.
[8] P. De, E.J. Dunne, J.B. Ghosh, and C.E. Wells, *Complexity of the discrete time/cost trade-off problem for project networks*, Oper. Res. 45 (1997), pp. 302–306.
[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A fast an elitist multi-objective genetic algorithm: NSGA-II*, IEEE Trans. Evolut. Comput. 6 (2002), pp. 182–197.
[10] E. Demeulemeester, B. De Reyck, B. Foubert, W. Herroelen, and M. Vanhoucke, *New computational results on the discrete time/cost trade-off problem in project networks*, J. Oper. Res. Soc. 73 (1998), pp. 1153–1163.
[11] E. Demeulemeester and W. Herroelen, *Project Scheduling*, Kluwer Academic Publishers, Dordrecht, 2002, pp. 70–95.
[12] E. Demeulemeester, M. Vanhoucke, and W. Herroelen, *RanGen: A random network generator for activity-on-the-node networks*, J. Sched. 6 (2003), pp. 17–38.
[13] B. De Reyck and W. Herroelen, *A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations*, Eur. J. Oper. Res. 111 (1998), pp. 152–174.
[14] B. De Reyck and W. Herroelen, *An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations*, Comput. Oper. Res. 25 (1998), pp. 1–17.
[15] B. De Reyck and W. Herroelen, *The multi-mode resource-constrained project scheduling problem with generalized precedence relations*, Eur. J. Oper. Res. 119 (1999), pp. 538–556.
[16] K. El-Rayes and A. Kandil, *Time–cost–quality trade-off analysis for highway construction*, J. Constr. Eng. Manage. 131 (2005), pp. 477–486.
[17] L. Eriksson, E. Johansson, C. Kettaneh-Wold, and S.W. Wold, *Design of Experiments, Principles and Applications*, MKS Umetrics AB, Umea, 2008.
[18] S. Hartmann and D. Briskorn, *A survey of variants and extensions of the resource-constrained project scheduling problem*, Eur. J. Oper. Res. 207 (2010), pp. 1–14.
[19] W. Herroelen, B. De Reyck, and E. Demeulemeester, *Resource-constrained project scheduling: A survey of recent developments*, Comput. Oper. Res. 25 (1998), pp. 279–302.
[20] W. Herroelen and R. Leus, *Project scheduling under uncertainty: Survey and research potentials*, Eur. J. Oper. Res. 165 (2005), pp. 289–306.
[21] H. Iranmanesh, M. Skandari, and M. Allahverdiloo, *Finding Pareto optimal front for the multi-mode time, cost quality trade-off in project scheduling*, World Acad. Sci., Eng. Technology 38 (2008), pp. 512–516.
[22] H. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling and Control*, John Wiley & Sons, Hoboken, NJ, 2009.
[23] K. Khalili-Damghani, A.R. Abtahi, and M. Tavana, *A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems*, Reliab. Eng. Syst. Safety 111 (2013), pp. 58–75.
[24] K. Khalili-Damghani, A.R. Abtahi, and M. Tavana, *A decision support system for solving multi-objective redundancy allocation problems*, Qual. Reliab. Eng. Int. 30 (2014), pp. 1249–1262.
[25] K. Khalili-Damghani and M. Amiri, *Solving binary-state multi-objective reliability redundancy allocation series-parallel problem using efficient epsilon-constraint, multi-start partial bound enumeration algorithm, and DEA*, Reliab. Eng. Syst. Safety 103 (2012), pp. 35–44.
[26] K. Khalili-Damghani, M. Nojavan, and M. Tavana, *Solving fuzzy multidimensional multiple-choice Knapsack problems: The multi-start partial bound enumeration method versus the efficient epsilon-constraint method*, Appl. Soft Comput. 13 (2013), pp. 1627–1638.
[27] K. Khalili-Damghani, M. Tavana, and S. Sadi-Nezhad, *An integrated multi-objective framework for solving multi-period project selection problems*, Appl. Math. Comput. 219 (2012), pp. 3122–3138.
[28] D. Khang and Y. Myint, *Time, cost and quality trade-off in project management: A case study*, Int. J. Project Manag. 17 (1999), pp. 249–256.
[29] J.Y., Kim, C.W. Kang, and I.K. Hwang, *A practical approach to project scheduling: Considering the potential quality loss cost in the time–cost tradeoff problem*, Int. J. Project Manag. 30 (2012), pp. 264–272.
[30] R. Kolisch and R. Padman, *An integrated survey of deterministic project scheduling*, OMEGA: Int. J. Manag. Sci. 29 (2001), pp. 249–272.
[31] O. Kramer, B. Gloger, and A. Goebels, *An experimental analysis of evolution strategies and particle swarm optimisers using design of experiments*, Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM Press, New York, 2007, pp. 674–681.
[32] T.S. Kyriakidis, G.M. Kopanos, and M.C. Georgiadis, *MILP formulations for single- and multi-mode resource-constrained project scheduling problems*, Comput. Chem. Eng. 36 (2012), pp. 369–385.
[33] G. Mavrotas, *Effective implementation of the epsilon-constraint method in multi-objective mathematical programming problems*, Appl. Math. Comput. 213 (2009), pp. 455–465.
[34] A.A. Najafi, S.T. Akhavan Niaki, and M. Shahsavar, *A parameter-tuned genetic algorithm for the resource investment problem with discounted cash flows and generalized precedence relations*, Comput. Oper. Res. 36 (2009), pp. 2994–3001.
[35] K. Neumann, C. Schwindt, and J. Zimmermann, *Project Scheduling with Time Windows and Scarce Resources*, 2nd ed., Springer, Berlin, 2003.

[36] B. Pollack-Johnson and M. Liberatore, *Incorporating quality considerations into project time/cost tradeoff analysis and decision making*, IEEE Trans. Eng. Manag. 53 (2006), pp. 534–542.
[37] S. Quintanilla, Á. Pérez, P. Lino, and V. Valls, *Time and work generalised precedence relationships in project scheduling with pre-emption: An application to the management of service centres*, Eur. J. Oper. Res. 219 (2012), pp. 59–72.
[38] M. Rahimi and H. Iranmanesh, *Multi objective particle swarm optimization for a discrete time, cost and quality trade -off problem*, World Appl. Sci. J. 4 (2008), pp. 270–276.
[39] M. Ranjbar, M. Khalilzadeh, F. Kianfar, and K. Etminani, *An optimal procedure for minimizing total weighted resource tardiness penalty costs in the resource-constrained project scheduling problem*, Comput. Ind. Eng. 62 (2012), pp. 264–270.
[40] M. Sakawa, *Fuzzy Sets and Interactive Multiobjective Optimization*, Plenum, New York, 1993.
[41] J.D. Schaffer, *Multiple objective optimization with vector evaluated genetic algorithms*, Proceedings of the First International Conference on Genetic Algorithms, 1985, pp. 93–100.
[42] N. Shahsavari Pour, M. Modarres, M. Aryanejad, and R. Tavakolli-Moghaddam, *The discrete time–cost–quality trade-off problem using a novel hybrid genetic algorithm*, Appl. Math. Sci. 4 (2010), pp. 2091–2094.
[43] Y. Shi and R. Eberhart, *Empirical study of particle swarm optimization*, in *Congress on Evolutionary Computation (CEC'1999)*, IEEE Press, Piscataway, NJ, 1999, pp. 1945–1950.
[44] W. Stadler, *Fundamentals of multicriteria optimization*, in *Multicriteria Optimization in Engineering and in the Sciences*, W. Stadler ed., Plenum Press, New York, 1988, pp. 1–25.
[45] H. Tareghian and S. Taheri, *On the discrete time, cost and quality trade-off problem*, Appl. Math. Comput. 181 (2006), pp. 1305–1312.
[46] H. Tareghian and S. Taheri, *A solution procedure for the discrete time, cost and quality tradeoff problem using electromagnetic scatter search*, Appl. Math. Comput. 190 (2007), pp. 1136–1145.
[47] M. Tavana, A.R. Abtahi, and K. Khalili-Damghani, *A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems*, Expert Syst. Appl. 41 (2014), pp. 1830–1846.
[48] M. Tavana, K. Khalili-Damghani, and A.R. Abtahi, *A new variant of fuzzy multi-choice Knapsack for project selection problem*, Ann. Oper. Res. 206 (2013), pp. 449–483.
[49] P.K. Tripathi, S. Bandyopadhyay, and S. Kumar Pal, *Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients*, Inf. Sci. 177 (2007), pp. 5033–5049.
[50] J. Weglarz, J. Jozefowska, M. Mika, and G. Waligora, *Project scheduling with finite or infinite number of activity processing modes a survey*, Eur. J. Oper. Res. 208 (2010), pp. 177–205.
[51] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*, Springer, London, 2010.
[52] H. Zhang and F. Xing, *Fuzzy-multi-objective particle swarm optimization for time–cost–quality tradoff in construction*, Autom. Constr. 19 (2010), pp. 1067–1075.
[53] A. Zhou, B. Qu, H.Z. Li, P. Suganthan, and Q. Zhang, *Multiobjective evolutionary algorithms: A survey of the state of the art*, Swarm Evol. Comput. 1 (2011), pp. 32–49.