Computers & Operations Research 50 (2014) 80–96

Contents lists available at ScienceDirect

Computers & Operations Research

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

A Multi-Objective Imperialist Competitive Algorithm for solving discrete time, cost and quality trade-off problems with mode-identity and resource-constrained situations

Elham Nabipoor Afruzi*, Amir Abbas Najafi[1], Emad Roghanian[2], Mostafa Mazinani[2]

Faculty of Industrial Engineering, K.N. Toosi University of Technology, Tehran 1999143344, Iran

ARTICLE INFO

Available online 18 April 2014

Keywords:
Project scheduling
Discrete Time–Cost–Quality Trade-off
Mode identity
Resource-constrained
MOICA

ABSTRACT

The Discrete Time–Cost–Quality Trade-off Problem (DTCQTP) is one of the most important problems in project scheduling applications. In DTCQTP an optimal combination of construction methods is decided with the objective of minimizing cost and time while maximizing quality. Due to real world resource-constrained situations in projects, this model is expanded to mode-identity resource-constrained DTCQTP model. In this model, each activity has multiple modes to be performed. In each mode, the required resources for performing the activity are different in at least one type. In addition, each activity can be done either in a normal way or in a crashing way in each performing mode. Based on the mode-identity situation, the activities belonging to the same subset should be executed in the same mode. In this paper, we present a Multi-Objective Imperialist Competitive Algorithm (MOICA) to solve our proposed model. We explain the elements of the algorithm and solve some problems generated for this model – including large size and small size instances – by this algorithm. The performance of our proposed algorithm is evaluated by comparison with four well-known algorithms: NSGAII; PESAII-clustering; PESAII-grid based; and SPEA2. The obtained results show the effectiveness of the proposed algorithm.

1. Introduction and literature review

Project management plays an important role in modern enterprise management which is defined as the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements [1]. The Resource-Constrained Project Scheduling Problem (RCPSP) is a common problem which researchers have devoted considerable effort for its investigation over the past decade. In modern enterprises, a large number of projects are set up to achieve the product innovation, and the resources which are used in projects are mostly manpower resources, which belong to renewable resources while the renewable resources have not attained sufficient importance [2]. This paper investigates the project scheduling problem under renewable resource-constrained condition.

Traditional resource-constrained project scheduling approaches have been restricted to the case where each activity may be performed in only one predefined way [3]. Thereafter some studies have investigated the Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPSP) where the job durations are discrete functions of job performance modes. In detail, in the multi-mode case, there are multiple modes for performing of activity i and each activity has to be performed in one out of M_i modes which may not change while the activity is executed. Also each mode represents an alternative combination of resource requirements of the activity and its duration. It is noticeable that in addition to the project scheduling, the selected mode for each activity should be determined at the end of problem solving process.

In MRCPSP, all mode-activity assignments are mutually independent and the activities do not force each other to be processed in a specific mode. In practice, however, situations may occur in which certain activities belong together and must be executed in the same mode. For example consider the assembling of the distillation tower and its joints stress relieving activities as the two main activities involved in the “distillation tower manufacturing” project. Suppose that there are two modes (methods) for performing of each activity in this project (for these two activities: horizontal assembling as mode 1 and vertical assembling as mode 2 for assembling activity and heat furnace method as mode 1 and electrical method as mode 2 for stress relieving activity). When the assembling activity of the distillation tower is done horizontally (mode 1), the stress relieving activity is usually done by the

* Corresponding author. Tel.: +98 21 84063340; fax: +98 21 88674858.
E-mail addresses: enabipoorafruzi@mail.kntu.ac.ir (E. Nabipoor Afruzi), aanajafi@kntu.ac.ir (A.A. Najafi), e_roghanian@kntu.ac.ir (E. Roghanian), mazinani.ie@gmail.com (M. Mazinani).
[1] Tel.: +98 21 84063378; fax: +98 21 88674858.
[2] Tel.: +98 21 84063348; fax: +98 21 88674858.

http://dx.doi.org/10.1016/j.cor.2014.04.003
0305-0548/© 2014 Elsevier Ltd. All rights reserved.

heat furnace (mode 1), but when the assembling activity of the distillation tower is done vertically (mode 2), the stress relieving activity is usually done by electrical element around the weld seams (mode 2). So, these two activities can be grouped in the same subset in the mentioned project and should be executed in the same mode. As another example, in the "rice cultivation" project, rice planting and rice transplanting are the two activities that can be grouped in the same subset. When the rice planting activity is done serially on the wooden board (mode 1), the rice transplanting activity is usually done by the transplanting machine automatically (mode 1), but when the rice planting activity is done in the mud – directly on the ground – (mode 2), the rice transplanting activity is usually done by hand (mode 2).

The case of mode identity problem was investigated for the first time by Salewski et al. [3] where the set of all jobs is partitioned into a number of disjoint subsets and all jobs belonging to the same subset have to be processed in the same mode. In this paper, the Mode Identity Resource-Constrained Project Scheduling Problem (MIRCPSP) is investigated to make the model as close as possible to real world projects.

Among RCPSPs, the Discrete Time–Cost Trade-off Problem (DTCTP) is a well-known problem in the project scheduling theory and applications where the duration of each activity is a discrete, no-increasing function of the amount of a single nonrenewable resource committed to it. Hindelang and Muth [4] introduced this problem for the first time. The Discrete Time–Cost Trade-off Problem is a strongly NP-hard optimization problem for general activity networks [2,5–7].

As a matter of fact, the DTCTP is a well-known problem in the project management literature. Given a set of modes (time–cost pairs) for each activity, the objective of the Discrete Time–Cost Trade-off Problem is to select a mode for each activity so that the total cost becomes minimized while meeting a given project deadline. Many methodologies have been proposed for Time–Cost Trade-off Problems to balance the completion time and cost by deciding an optimal combination of construction methods for all of the activities. These methodologies can be classified into three categories: (a) exact algorithms such as linear programming, integer programming, dynamic programming, branch and bound algorithms, etc. [8–11]; (b) heuristic algorithms [12–15]; (c) meta-heuristic algorithms [2,16–18,34].

As mentioned before, there have been extensive studies on Time–Cost Trade-off Problem while recent contracts consider the quality performance of projects in addition to time and cost [19]. So, considering the quality objective in addition to time and cost objectives is an important necessity. But unfortunately few researchers investigate this problem in their studies [19–23]. This paper considers three objectives: time; cost; and quality as a DTCQTP to overcome this shortcoming.

The other considerable point is that in practice situations may occur in which some activities might be crashed to get a shortened duration by more direct costs. In other words, each activity in each method can be performed in a crashing way instead of its normal way with less time and more direct costs. Many researchers have considered this fact in their studies. In this paper, both normal and crashing ways are considered for activity performance in each mode in order to consider real world situation. It is necessary to mention that in this paper the term "mode" represents "method" of activity performing and the resources are different in at least one type in each mode of the activity performance.

In this paper, a Mode Identity Resource-Constrained Discrete Time–Cost–Quality Trade-off Model (MIRC-DTCQTP) is proposed that has some characteristics as follows:

(i) The splitting of the activity is not admissible.
(ii) The durations of the activities are deterministic.
(iii) The renewable resources are focused which are constrained in each time period.
(iv) Each activity is executed not only in a selected mode, but also in a selected way, which determines whether the activity is executed in the crashing way to get a shortened duration by more direct costs.
(v) The direct cost per unit time of resource k is dependent on three factors: activity; the method by which the activity is performing; and the way in which the activity is performed (crashing way or normal way). This assumption is proposed to cover some shortcomings in existing models which assume the cost of resource k is equal in all ways and methods the activity is performed. For example, in many industrial projects the welding activity can be done either by Tig welding method or the Shielded Metal Arc Welding (SMAW) method. The different welder costs per unit time for welding activity in Tig welding method and SMAW method is an example of resource direct cost per unit time dependency on method. Another example for showing resource direct cost per unit time dependency on activity is the different costs of milling machine for gear wheel manufacturing activity and rough milling activity.
(vi) The units of resource k required for performing activity i, if it is executed by method m, are relying on the way (normal or crashing) of the activity execution. It is apparent that in the crashing way, the required resources cannot be less than the quantity required for the normal way.
(vii) For activity i, the required resources in each possible performing method, are different in at least one type in comparison with the other methods. In other words, in each method, the quantities of the required resources can be changed based on the normal performing way or crashing performing way, but the types of the required resources cannot be changed. However, by changing the activity performing method, it is obvious that the types of the required resources should be different in comparison with the other methods.
(viii) The quality of activity i is dependent on two factors: the activity performance method (mode) and the activity execution way. Obviously for each activity in each method, the activity quality in the normal way is better than the activity quality in the crashing way.
(ix) All the activities in the same subset should be performed in the same mode according to the mode identity problem situation.

The aim of this research is to obtain the best method for performing of the activities in each subset and the best way of performing (normal or crashing) for each activity in order to minimize the project time, minimize the project cost, and maximize the project quality under resource and mode identity constraints. According to the fact that the MIRC-DTCQT problem has three objective functions, in this paper, we present the Multi-Objective Imperialist Competitive Algorithm (MOICA) for solving this problem and the obtained results are compared with four other multi-objective meta-heuristic algorithms: Non-dominated Sorting Genetic Algorithm (NSGAII) [24], Pareto Envelope-based Selection Algorithm-clustering (PESAII), Pareto Envelope-based Selection Algorithm-grid Based (PESAII) [25,26], and Strength Pareto Evolutionary Algorithm (SPEA2) [27].

The structure of the paper is as follows: Section 2 describes the problem and its mathematical model. The proposed algorithm, MOICA, is explained in detail in Section 3. Section 4 shows the experimentation and computational results, and finally, the conclusion and further research are provided in Section 5.

2. Problem explanation and mathematical formulation

As detailed explanation of the introduction, the real world projects have many features to which the necessary attention has not been devoted in the research up to now. This paper tries to overcome this shortcoming by considering some of the most important features of real world projects in the mathematical model.

In this paper, for the first time, the multi objective time, cost, quality trade-off problem under resource and mode identity constraints is studied. So in the proposed model, each activity has more than one method for performing while it has to be performed in one out of its possible methods which may not change while the activity is executed. It is noticeable that according to the mode identity situation, definite activities are dependent from each other and must be executed in the same mode. Also in each mode, the normal way and the crashing way are the two alternatives for activity execution. According to the objective functions, simultaneous minimization of the project time and the project cost and maximization of the project quality is desirable.

The structure of the project is depicted by activity-on-node (AON) network $G=(V,E)$ where the nodes represent the activities $i=1,2,3,\dots,n$ and the arcs represent the precedence relations (E). The activity i can be started after all its predecessors (P_i) are finished. There is a start activity in the network and activity n is the finish activity. According to the mode identity constraint, the set of project activities is partitioned into U disjointed subsets while all activities forming one subset have to be processed in the same mode. Also in each method (mode), an activity i can be performed either in the normal way or the crashing way. Obviously for each activity in each method, the duration of the activity, the resource k requirements per unit time, the direct cost per unit time of resource k for activity performance, and the activity quality are totally relying on the way (normal or crashing) of activity execution. Based on the resource constrains, there are $k \in K$ types of renewable resource where r_k equals to the available resource type k for each time period as an upper bound of resource k capacity and it is fixed in each time period.

In this part, the mathematical formulation of the Mode Identity Resource-Constrained Discrete Time–Cost–Quality Trade-off Problem is presented. Section 2.1 describes the notation of indices, parameters, and variables used in the model, Section 2.2 presents the mathematical model, and Section 2.3 explains the presented model.

2.1. Model notations

The notations used in the model are as follows:
Indices

V	the set of activity nodes
T	the set of time periods
U	the set of disjointed subsets of activities
K	he set of renewable resources
E	the set of precedence relations between activities
M_i	the set of activity i modes

Parameters

d^e_{im}	the crashing duration of activity i performing in mode m
c^e_{imk}	the per unit time direct cost of resource k to perform activity i in the crashing way and mode m
r^e_{imk}	the per unit time usage of resource k required for activity i if it is executed in the crashing way and mode m
d^n_{im}	the normal duration of activity i performing in mode m
c^n_{imk}	the per unit time direct cost of resource k to perform activity i in the normal way and mode m
r^n_{imk}	the per unit time usage of resource k required for activity i if it is executed in the normal way and mode m
r_k	the available quantity of resource type k in each time period
H_u	specific nonempty subset u of activities
f_u	the activity with the smallest index in subset H_u
w_i	qualitative weight of activity i
Q^e_{im}	the quality of activity i performing in the crashing way and mode m
Q^n_{im}	the quality of activity i performing in the normal way and mode m

Variables

Y_i	a decision variable with value one when activity i is performed in the crashing way and it is zero when activity i is performed in the normal way
X_{imt}	a decision variable with value one when activity i is performed in mode m and completed in period t
$F_n=Z_2$	the total project time which is equal to the finish time of activity n
F_j	the finish time of activity j
Z_1	the total project direct cost
Z_3	the total project quality

2.2. Problem modeling

The three objective functions of our model are presented in (1)–(3).

$$\min Z_1 = \sum_{i \in v} \sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} \left[\left(Y_i * X_{imt} * \sum_{k \in K} (d^e_{im} * c^e_{imk} * r^e_{imk}) \right. \right.$$
$$\left. \left. + (1-Y_i) * X_{imt} * \sum_{k \in K} (d^n_{im} * c^n_{imk} * r^n_{imk}) \right) \right] \tag{1}$$

$$\min Z_2 = \sum_{t=EF_n}^{LF_n} t * x_{n1t} \tag{2}$$

$$\max Z_3 = \sum_{i \in v} W_i * \sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} [(Y_i * X_{imt} * Q^e_{im}) + ((1-Y_i) * X_{imt} * Q^n_{im})] \tag{3}$$

The constraints of our model are shown in (4)–(9) s.t.

$$\sum_{m \in M_{f_u}} \sum_{t=EF_{f_u}}^{LF_{f_u}} X_{f_u mt} = 1 \quad (1 \le u \le U), \quad f_u = \min j \in H_u \tag{4}$$

$$\sum_{t=EF_j}^{LF_j} X_{jmt} = \sum_{t=EF_{f_u}}^{LF_{f_u}} X_{f_u mt}$$
$$(1 \le u \le U, \quad \forall j \in H_u, \quad \forall m \in M_j) \tag{5}$$

$$\sum_{m \in M_j} \sum_{t=EF_j}^{LF_j} \Big(t - [(d^e_{jm} * Y_j) + (d^n_{jm} * (1-Y_j))] \Big) * X_{jmt}$$
$$\ge \sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} t * X_{imt}, \quad \forall (i,j) \in E \tag{6}$$

$$\sum_{i \in v} \sum_{m \in M_i} ((Y_i * r^e_{imk} + ((1-Y_i) * r^n_{imk})) \sum_{l=\ \max\{t, Ef_i\}}^{\min\{t+[(Y_i.d^e_{im})+((1-Y_i).d^n_{im})]-1,\ Lf_i\}}$$
$$X_{imt} \le r_k, \quad \forall k \in K \tag{7}$$

$$X_{imt} = \{0,1\} \quad \forall i \in V, \ \forall m \in M_i, \ \forall t \in T \tag{8}$$

$$Y_i = \{0,1\} \quad \forall i \in V \, . \tag{9}$$

2.3. Model explanation

The objective function (1) in the mathematical model indicates the total direct cost of the project. The direct cost has two sections: if the activity i executed in the crashing way $(Y_i = 1)$ and performed in mode m and completed in period t $(X_{imt} = 1)$, the $\sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} (Y_i . X_{imt} . \sum_{k \in K} (d^e_{im} . c^e_{imk} . r^e_{imk}))$ results the direct cost of activity i while the second section $\sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} [(1-Y_i).X_{imt}.\sum_{k \in K}(d^n_{im}.c^n_{imk}.r^n_{imk})] = 0$ since $(1-Y_i) = 0$.

If the activity i executed in the normal way $(1-Y_i = 1)$ and mode m and completed in period t $(X_{imt} = 1)$ the second section $\sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} [(1-Y_i).X_{imt}.\sum_{k \in K}(d^n_{im}.c^n_{imk}.r^n_{imk})] = 0$ results the direct cost of activity i and the first section $\sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} (Y_i.X_{imt}.\sum_{k \in K}(d^e_{im}.c^e_{imk}.r^e_{imk})) = 0$ since $(Y_i = 0)$.

The summation of these quantities (which obtained for each activity) for all of the activities, leads to the total project direct cost that should be minimized as the first objective of our problem.

The objective function (2) is the total project time and should be minimized.

The objective function (3) shows the total project quality which should be maximized. If the activity i executed in the crashing way $(Y_i = 1)$ and mode m and completed in period t $(X_{imt} = 1)$, formula $\sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} (Y_i.X_{imt}.Q^e_{im})$ represents the quality of activity i while the second section is equal to zero since $(1-Y_i) = 0$. If the activity i executed in the normal way $(1-Y_i = 1)$ and mode m and completed in period t $(X_{imt} = 1)$, formula $\sum_{m \in M_i} \sum_{t=EF_i}^{LF_i} ((1-Y_i).X_{imt}.Q^n_{im})$ shows the quality of activity i while the first section is equal to zero since $(Y_i = 0)$. The summation of these two parts represents the quality of activity i which should be multiplied by the weight of the activity i in order to obtain the quality of the activity i according to the project. By adding the value of all activities quality together, the total quality of project can be obtained as shown in Eq. (3).

Eq. (4) ensures that each activity is being performed only in one mode. Eq. (5) is the mode identity constraint that states that all activities forming one subset have to be performed in the same mode.

Constraint (6) shows the precedence relationships in which relying on the performing way of the activity j $(Y_j = 0 \text{ or } 1)$, $\sum_{m \in M_j} [(d^e_{jm}.Y_j) + (d^n_{jm}.(1-Y_j))]$ shows the duration of activity j.

In constraint (7), the quantity of the resource k required for performing of activity i when executed in mode m and completed in period t $(x_{imt} = 1)$, can be obtained by $\sum_{m \in M_i} (Y_i.r^e_{imk}) \sum_{l=\max\{t,Ef_i\}}^{\min\{t+(Y_i.d^e_{im})-1, Lf_i\}} X_{imt}$ when the activity is performed in the crashing way $(Y_i = 1)$ or it can be obtained by $\sum_{m \in M_i} ((1-Y_i).r^n_{imk}) \sum_{l=\max\{t,Ef_i\}}^{\min\{t+((1-Y_i).d^n_{im})-1, Lf_i\}} X_{imt}$ when the activity is performed in the normal way $(1-Y_i = 1)$. The summation of these quantities for all activities of the project results the required amount of resource k in period t that should be less than r_k which is the available resource type k in each time period. This constraint should be true in each time period $[t-1, t]$ and for each of the K resources.

Eqs. (8) and (9) describe the binary variables X_{imt} and Y_i.

3. Multi-Objective Imperialist Competitive Algorithm (MOICA)

Multi-Objective Imperialist Competitive Algorithm is the proposed algorithm which is used in this paper to solve the mode-identity resource-constrained Discrete Time–Cost–Quality Trade-off Problem. Before the step by step description of the proposed algorithm, the encoding and decoding of a solution should be devised. Also the schedule generation algorithm which assigns a figure of merit to each encoded solution as a fitness function is required.

3.1. Encoding and decoding of a solution

Each solution can be shown as a triplet $1 \times N$ matrix in which N is the number of project activities. This triplet matrix describes the three characteristics of the project activities: the ordered values of the activities; the selected modes; and the selected ways. Fig. 1 shows the encoding of a solution for the project with 7 activities, as an example, where there are 3 modes for performing of each activity and the project has 3 subsets as shown in Table 1.

The first part of the solution describes the list of order of the activities that shows the order of the activities entrance in scheduling process. In this example, Fig. 1 shows that activity 3 is the first activity in the list of order of the activities, in other words, the activity 3 is the first activity that should be scheduled on the basis of the scheduled generation algorithm explained in Section 3.2. It is important to note that this activity can be started only when all of its predecessors are finished and enough resources are available for executing it; otherwise, the next activity in the list of order of the activities, activity 4 in this example, should be considered and so on.

The second part of the solution represents the selected modes of the respective activity in the first part. For instance, activity 3 is executed by mode 1. According to the fact that in mode identity problem, the activities forming the same subset should be performed in the same mode (method), the code refining is needed here. For this purpose, we find all activities which are in the same subset of activity 3; i.e., {1,3,7} and select one of these activities randomly and assign its mode to all activities in its subset. For example, if activity 7 is selected, we assign mode 2 to all the activities {1,3,7}. By applying this refining method, one of the possible encoding for this project is presented in Fig. 2.

The third part of the solution depicts the selected way in which the activity is done. For example, activity 3 is done in the crashing way in the above representation which is shown by number 1 and the activity 4 is done in the normal way shown by number 0.

3.2. Schedule generation algorithm

The fitness of each solution is determined by evaluating its performance with respect to the three objective functions which

3	**4**	**5**	**2**	**1**	**7**	**6**	List of order of activities
1	**2**	**3**	**1**	**3**	**2**	**2**	The selected modes
1	**0**	**1**	**1**	**0**	**1**	**0**	The selected ways

Fig. 1. The structure of the solution matrix.

Table 1
Partitioning the set of all jobs in the mode identity case.

The set of disjointed subsets of activities (U)	Specific nonempty subset u of activities (H_u)	The activity with the smallest index in subset H_u (f_u)
1	{1,3,7}	1
2	{2,6}	2
3	{4,5}	4

are presented in Section 2.1 as the minimization of the project total cost, minimization of the project total time, and maximization of project total quality.

Before explanation of the schedule generation algorithm, it seems necessary to introduce the notations used in this part:

PS	the scheduled activities matrix
P	activities predecessors matrix
P_j	predecessors of activity *j*
US	the matrix of unscheduled activities which their predecessors finished completely
j^*	the activity by premier order in matrix *US*
ES_j	the activity by premier order in matrix *US*
f_j	finish time of activity *j*
S_j	the start time of activity *j*

In this algorithm, the scheduling process of the activities is divided into "*n activity*" stages while in each stage, one activity is scheduled. For this purpose, based on the encoding of the problem, three matrices: the activities order; the activities methods; and the activities ways are taken into consideration. The first activity in the list of order of the activities which has no

3	4	5	2	1	7	6	List of order of activities
2	2	2	1	2	2	1	The selected modes
1	0	1	1	0	1	0	The selected ways

Fig. 2. The refined structure of the solution matrix.

predecessors will be scheduled in the first stage. Then the other activities based on their location in the list of order of the activities will be scheduled in the next stages. The activities that are scheduled will be placed at the matrix (*PS*). In detailed explanation, in this scheduling process, with respect to the predecessors' matrix of the activities (*P*), the activities which are not scheduled yet and all of their predecessors finished completely till that stage will be reserved in matrix (*US*). In each stage, one of the activities of the *US* matrix which has the first position amongst the activities (it is located before other activities) in the list of order of the activities, will be scheduled (j^*). For scheduling of an activity, first of all its earliest start time (*ES*) will be calculated. The activity's *ES* equals to the maximum finish times (*f*) of its predecessors. If there is no predecessor for an activity, then its *ES* is equal to zero (the outset of the scheduling time horizon). Once the activity's *ES* has been calculated, the required resources for performing of the activity can be examined with respect to its method and its way. If the required resources are available during the time of activity execution, the start time of the activity (*S*) will be the earliest start time of the activity (*ES*); otherwise it will be transferred to the earliest time to which the required resources can be devoted. After determining the start time of an activity, by adding its duration, the finish time of the activity will be attained. Also, in this stage, the activity cost will be computed according to its duration, applied resources, and per unit time direct cost of the resources. Furthermore, the quality of the activity will be determined according to its method and way of executing. Finally, by performing all of the activities, three mentioned objective functions; total time of the project, total cost of the project, and the quality of the whole project will be calculated.

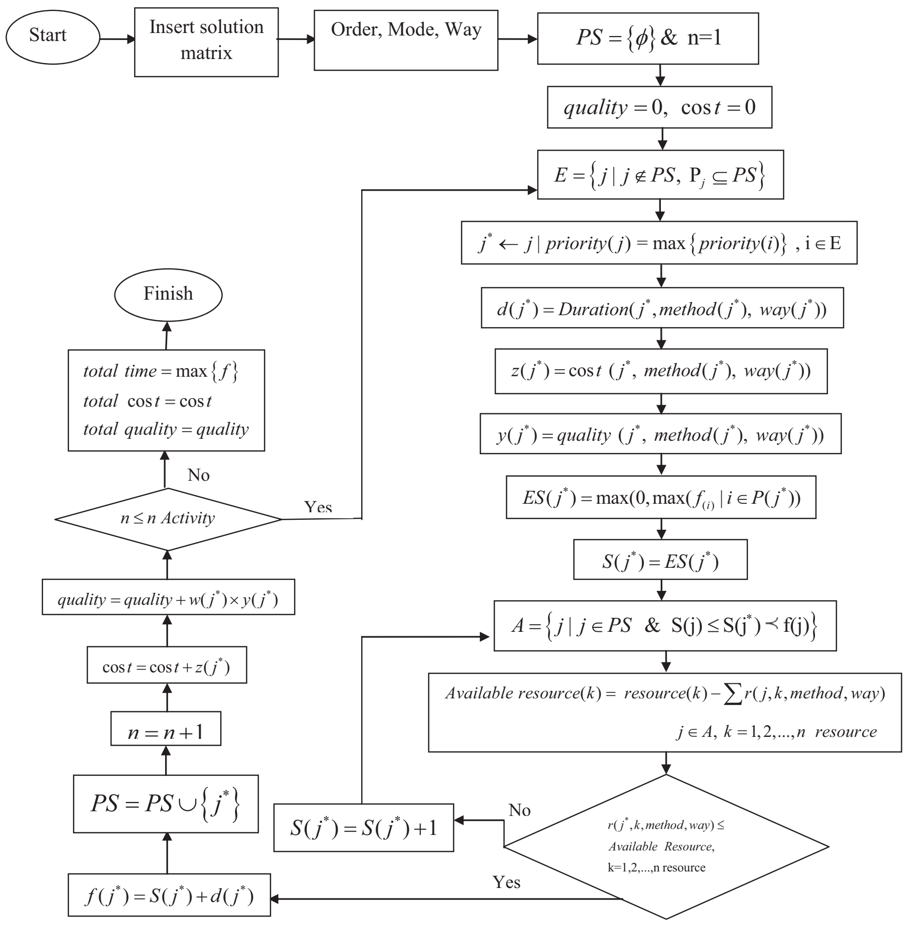
Start
Insert solution matrix
Order, Mode, Way
$PS=\{\phi\}$ & n=1
$quality=0,\ cost=0$
$E=\{j\mid j\notin PS,\ P_j\subseteq PS\}$
$j^*\leftarrow j\mid priority(j)=\max\{priority(i)\},\ i\in E$
$d(j^*)=Duration(j^*,method(j^*),\ way(j^*))$
$z(j^*)=cost\ (j^*,\ method(j^*),\ way(j^*))$
$y(j^*)=quality\ (j^*,\ method(j^*),\ way(j^*))$
$ES(j^*)=\max(0,\max(f_{(i)}\mid i\in P(j^*))$
$S(j^*)=ES(j^*)$
$A=\{j\mid j\in PS\ \&\ S(j)\le S(j^*)\prec f(j)\}$
$Available\ resource(k)=\ resource(k)-\sum r(j,k,method,way)$ $j\in A,\ k=1,2,\dots,n\ resource$
$r(j^*,k,method,way)\le$ Available Resource, k=1,2,...,n resource
No
$S(j^*)=S(j^*)+1$
Yes
$f(j^*)=S(j^*)+d(j^*)$
$PS=PS\cup\{j^*\}$
$n=n+1$
$cost=cost+z(j^*)$
$quality=quality+w(j^*)\times y(j^*)$
$n\le n\ Activity$
Yes
No
$total\ time=\max\{f\}$
$total\ cost=cost$
$total\ quality=quality$
Finish

Fig. 3. Schedule generation schema.

The schedule generation algorithm is shown schematically in Fig. 3.

3.3. *Generation of initial empires*

Each solution acts as a "country" in Imperialist Competitive Algorithm (ICA). The term "country" is the same as the definition "chromosome" in GA terminology. In an N-dimensional optimization problem, a country is shown as an $1 \times N$ array. This array is defined by $country = [p_1, p_2, p_3, \ldots, p_N]$ where p_i is the variable in a country denotes a socio-political characteristic of a country and should be optimized. In this point of view, the algorithm searches for the best country, i.e., the country with the best combination of socio-political characteristics, such as culture, language, and economic policy [28]. After the generation of the countries as an initial population, the non-dominance technique [24,34] is used to shape the population in different Pareto fronts. In addition, the crowding distance technique [24,34] is applied to rank the members in each front. The members of the first Pareto front should be inserted into the archive; in which the solutions at the end of the algorithm contain the final set of non-dominated solutions. In detail, the role of the archive is keeping and updating the non-dominated solutions in each of the iterations.

3.4. *Imperialists selection*

The best solutions according to the non-dominance and crowding distance techniques are selected from population as the imperialists by size *nImp*. Thus, the remaining countries act as colonies. In order to assign the colonies to the imperialists, the power of each imperialist should be calculated by performing the following steps.

3.4.1. *Normalized cost of objective function i for imperialist n*

Each imperialist has a cost value which affects directly on the number of its colonies. For calculating the cost value of each imperialist, first the normalized cost of objective function i for imperialist n; $cost_{i,n}$should be calculated according to Eq. (10).

$$\cos\ t_{i,n} = \frac{|f_{i,n} - f_i^{worst}|}{|f_i^{max} - f_i^{min}|} \tag{10}$$

where $f_{i,n}$ is the value of objective function i for imperialist n, f_i^{worst} is the worst value of objective function i in population in each iteration. f_i^{max} and f_i^{min} are the maximum and minimum values of objective function i in each iteration, respectively.

3.4.2. *Normalized cost of imperialist n*

The normalized cost value of each imperialist, $Total\ cost_n$, is obtained by Eq. (11) where r is the number of objective functions.

$$Total\ cost_n = \sum_{i=1}^{r} \cos\ t_{i,n}. \tag{11}$$

3.4.3. *Imperialist power*

The power of each imperialist can be obtained by Eq. (12) where the nominator is the normalized cost value of the imperialist and the denominator is the summation of the normalized cost values for all of the imperialists.

$$p_n = \left| \frac{Total\ cost_n}{\sum_{i=1}^{N_{imp}} Total\ cost_n} \right| \tag{12}$$

3.4.4. *Colonies assignment to the imperialists*

Depending on the power of each imperialist, the colonies can be distributed around that imperialist. Eq. (13) shows the initial number of colonies (NC_n) that should be assigned to the nth imperialist.

$$NC_n = round\ \{P_n.N_{col}\} \tag{13}$$

where N_{col} is the total number of the colonies. We randomly select some colonies by size NC_n and assign them to the imperialist n. It is obvious that the more power the imperialist has, the greater the number of colonies can be assigned to it.

3.5. *Total power of an empire*

The total power of an empire is obtained by the power of its imperialist and the power of its colonies. The total power of an empire is mainly affected by the power of its imperialist country and the power of the colonies has a minor effect on it. Eq. (14) shows the total power of empire $n(TP\ Emp_n)$.

$$\begin{aligned} TP\ Emp_n = & (total\ cost(imperialist_n) \\ & + \xi\ mean\{total\ cost(colonies\ of\ empire_n)\})(QE_n) \end{aligned} \tag{14}$$

where ξ is a number between 0 and 1. The smaller the value of ξ, the lower the effect of colonies on the power of the empire and vice versa. It is noticeable that the role of imperialist on the power of the empire is more than the role of colonies. The calculation of colony total cost is like the calculation of imperialist total cost and can be obtained by Eq. (11). QE_n is the quality of the nth empire and can be obtained as follows: by putting all the imperialists and colonies together and selecting the non-dominated solutions, we can obtain the percentage of the non-dominated solution belonging to each empire that is considered as the quality of that empire.

3.6. *Moving the colonies of the empire toward the imperialist (assimilating)*

In each empire, the colonies move toward the imperialist in assimilating movement. In fact, the imperialist enhance its colonies in this operation. As shown in Fig. 4, for a project with 7 activities, we use a binary random array with $\beta = 0.3$. β is a number between 0 and 1 and shows the density of value "1" in binary random array. Wherever the binary random array is 1, the corresponding imperialist value should be assigned to the colony and wherever it is 0, the colony value remains as before. For assimilating movement, one part of the colony matrix or dual combinations or all the three parts of the colony matrix can be chosen. Fig. 4 shows the assimilating movement for each part of the colony matrix. It is noticeable that the activities are unique and cannot be repeated in the first part of the colonies matrix. Therefore, after assigning the imperialist arrays to the colony, the other

Imperialist	3	2	5	1	6	4	7	3	2	1	1	2	3	3	0	0	1	0	1	1	1
Colony	2	4	7	3	5	1	6	2	3	3	1	2	1	1	1	0	0	1	0	1	0
Binary array	0	1	0	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0
	4	2	7	1	3	5	6	2	3	1	1	2	3	1	0	0	0	1	1	1	0

Fig. 4. Moving colonies toward the imperialist (assimilating).

arrays come from the remaining activities in the colony in the previous order.

3.7. Information sharing between colonies

In each empire, colonies share their information with each other by crossover operation to enhance themselves. The population percentage that share information is shown by P_c. For crossover operation, one part, dual combinations, or a triple combination of the colony matrix can be chosen. The best colonies have more chance to share their information than the others because the colonies are selected by binary tournament selection technique which is explained as follows.

3.7.1. Binary tournament selection

In this technique, first, we should select two solutions from the population randomly. Then, if these solutions are from different fronts, the solution from the lower front number is selected; otherwise when they are from the same front, the solution with higher crowding distance should be selected. As a result, this technique helps the better solutions to have more chance to be selected.

Three types of crossover operators are applied in this paper for the colonies information sharing. These three types are as follows: one point crossover; two point crossover and uniform crossover. One of these operators will be chosen randomly for the information sharing of the selected part (or parts).

One point crossover operators are illustrated in Fig. 5 for each part of the triplet matrix of the colony.

Also the two point crossover and uniform crossover operators are depicted in Figs. 6 and 7 respectively for each part of the triplet matrix of the colony.

3.8. Revolution

In each decade, revolution operators are performed on some of the colonies in each empire. The population percentage that revolt is shown by P_r. For revolution operation, one part, dual combinations, or a triple combination of the colony matrix can be chosen. The colonies selection is done by binary tournament selection technique, so the best colonies have more chance to revolt.

Swap, insertion, reversion and perturbation are the four revolution operators which are applied in this paper. One of these operators will be chosen randomly for the revolution of the selected part (or parts). The swap operators are displayed in Fig. 8 for each part of the triplet matrix of the colony.

Also the insertion, reversion and perturbation operators for each part of the triplet matrix of the colony are illustrated in Figs. 9, 10, and 11 respectively.

3.9. Imperialist improvement

In each empire, the imperialist produce some neighbors according to the total power of its empire. The neighbors' production operators are the same as the revolution operators which are explained in Section 3.8. The imperialist in the empire with minimum total power produces minimum neighbors (Ne_{min}) and imperialist in the empire with the maximum total power produces maximum neighbors (Ne_{max}). The number of neighbors produced by each imperialist is dependent on the total power value of its empire and is calculated by using a linear function varying between Ne_{min} and Ne_{max} as shown in Eq. (15).

$$Ne_n = floor\left(Ne_{min} + \frac{(Ne_{max} - Ne_{min})(TP\,Emp_{worst} - TP\,Emp_n)}{(TP\,Emp_{worst} - TP\,Emp_{best})}\right) \tag{15}$$

where $TP\,Emp_{best}$ is the total power of the most powerful empire, $TP\,Emp_{worst}$ is the total power of the empire with the least power, and $TP\,Emp_n$ is the total power of the empire which we want to calculate the number of neighbors produced by its imperialist. Fig. 12 illustrates the relationship between the number of neighbors and the total power of the empire.

3.10. Colonies updating

In each empire, the initial colonies, population obtained by assimilating, population produced by information sharing, population obtained by revolution, and the neighbors from imperialist improvement should be merged together and the non-dominated solutions should be entered into the archive. Then the best colonies according to the non-dominating technique and crowding distance technique should be selected with size $NC(i)$ and the other members should be deleted.

3.11. Archive updating

As mentioned in Section 3.10, the merged population should be ranked according to the non-dominating and crowding distance techniques and the non-dominated solutions should be entered into the archive. The archive members are also ranked according to the non-dominating and crowding distance techniques and the solutions of the first front are reserved and the others are omitted. Also, the similar solutions should be deleted from the archive. If the number of these non-dominated solutions is more than the archive size, *narchive*, their crowding distance should be calculated and the solutions with more crowding distance should be chosen with size *narchive* and the other solutions should be omitted.

3.12. Exchanging the positions of the imperialist and a colony

First, the total cost of all imperialists should be updated. Next, in each empire, the imperialist and the colonies should be integrated together. Then, they should be sorted based on the

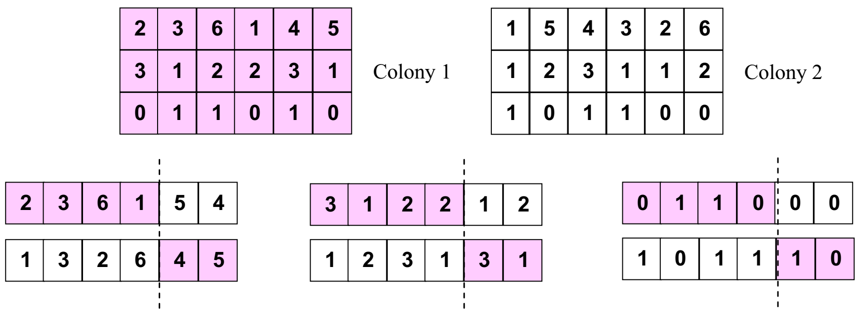

Fig. 5. Information sharing of each part by one point crossover operators.

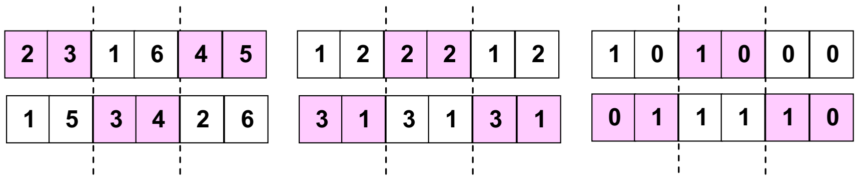

Fig. 6. Information sharing of each part by two point crossover operators.

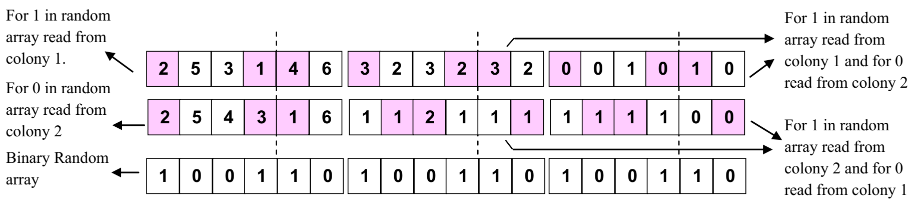
For 1 in random array read from colony 1.
For 0 in random array read from colony 2
Binary Random array
For 1 in random array read from colony 1 and for 0 read from colony 2
For 1 in random array read from colony 2 and for 0 read from colony 1

Fig. 7. Information sharing of each part by uniform crossover operators.

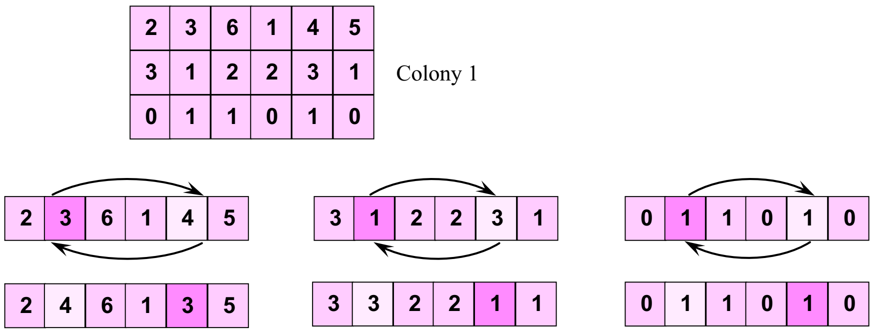

Fig. 8. Revolution of each part by swap operators.

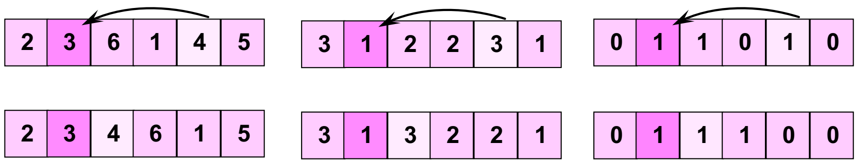

Fig. 9. Revolution of each part by insertion operators.

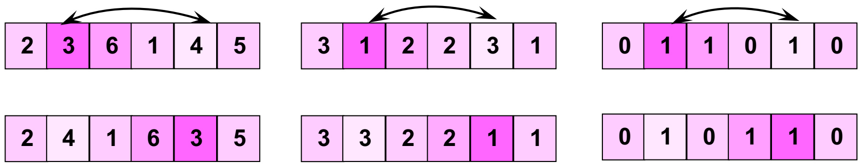

Fig. 10. Revolution of each part by reversion operators.

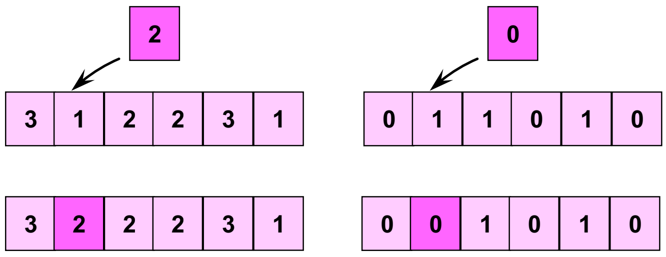

Fig. 11. Perturbation operators for revolution of selected mode and selected way parts.

non-dominating and crowding distance techniques. Finally, the best member should be selected as the imperialist. In other words, in each empire, the member with the highest total cost (best total cost) should be selected as the imperialist among the others which can be either the previous imperialist or the previous colony. Fig. 13 shows this position exchanging.

3.13. Imperialistic competition

In imperialistic competition, the power of the weak empires dwindles and the power of the powerful empires increases. All the empires compete with each other to possess the weakest colony of the weakest empire. It is noticeable that the most powerful empire is not the certain winner of the weakest colony, but it has only the most probability to possess the weakest colony of the weakest empire. In other words, the weakest colony of the weakest empire should be selected. Then, this colony should be assigned to the luckiest empire. To calculate the possession probability of each empire, the normalized total power of each empire should be

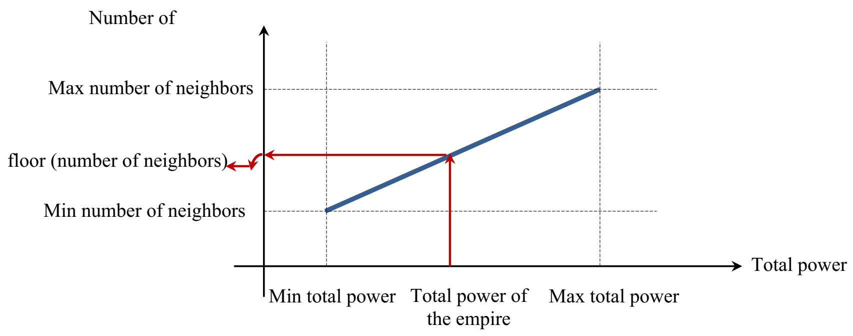

Fig. 12. The relationship between the total power of each empire and the number of its neighbors.

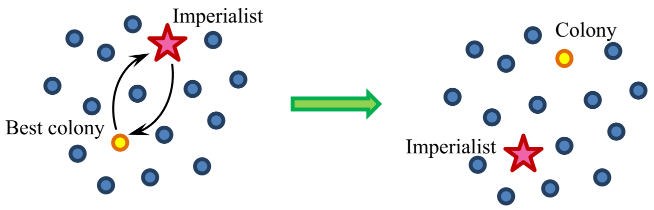

Fig. 13. Exchanging the positions of the imperialist and a colony.

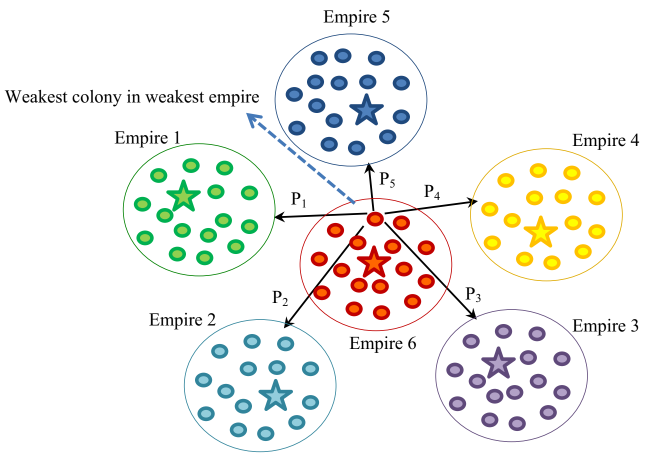

Fig. 14. Imperialistic competition.

determined according to Eq. (16).

$$NTP\ Emp_n = TP\ Emp_n - \min_i\{TP\ Emp_i\} \quad (16)$$

where $NTP\ Emp_n$ is the normalized total power of the nth empire and $TP\ Emp_n$ is the total power of the nth empire. Now, the possession probability of each empire can be calculated by Eq. (17).

$$p_{Pn} = \left| \frac{NTP\ Emp_n}{\sum_{i=1}^{N_{imp}} NTP\ Emp_i} \right| \quad (17)$$

Having the possession probability of each empire, the most fortunate empire will be selected by roulette wheel technique to win the weakest colony. Fig. 14 shows the imperialistic competition schematically.

3.14. Elimination of the powerless empire

The weak empires collapse gradually and their colonies should be distributed among the other empires. When an empire loses all

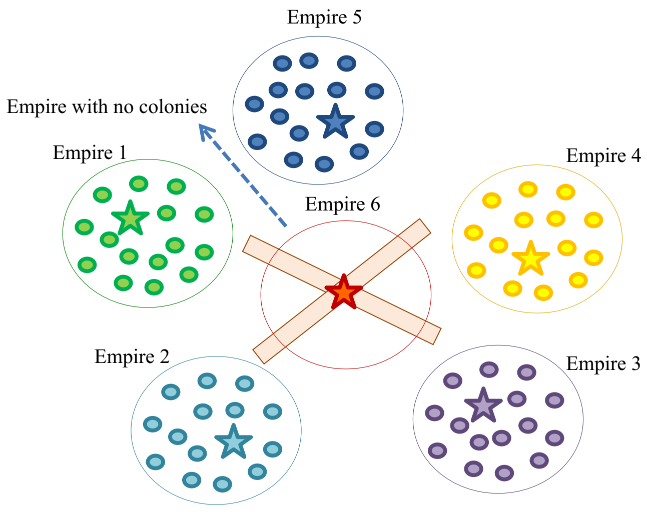

Fig. 15. Elimination of the powerless empire.

of the colonies belonging to it, this empire should be collapsed. Fig. 15 shows the elimination of the empire 6 schematically.

3.15. Stopping criterion

The stopping criterion is considered when there is only one empire that governs all of the colonies. Fig. 16 shows the MOICA schematically.

4. Experimentation and computational results

In this section, the evaluation of the proposed algorithm is studied. The performance of the proposed MOICA is validated by comparison with four well-known multi-objective algorithms; NSGAII [24], PESAII-clustering, PESAII-grid based [25,26], SPEA2 [27]. All the algorithms studied in this paper are coded using MATLAB 2009.b and run on a personal computer with a 2.20 GHz Intel Core 2 Duo CPU and 1 GB main memory.

4.1. Problem generation

In order to the fact that the studied problem is a newly defined one in some aspects, we cannot find any standard test problems (benchmark problem) for the presented mathematical model to examine the performance of the proposed algorithm introduced in this paper. Therefore, 25 standard test problems containing 13 small size problems and 12 large size problems from the project scheduling problem library *PSPLIB* site in the University of Kiel [29]

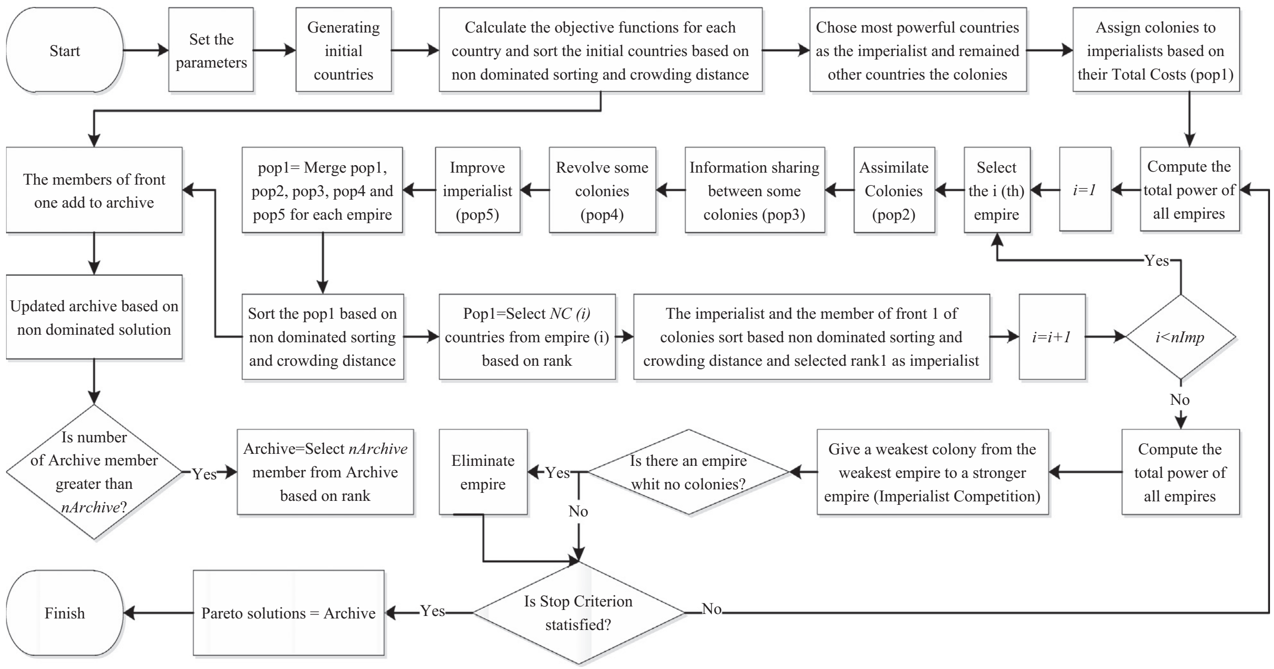
Start
Set the parameters
Generating initial countries
Calculate the objective functions for each country and sort the initial countries based on non dominated sorting and crowding distance
Chose most powerful countries as the imperialist and remained other countries the colonies
Assign colonies to imperialists based on their Total Costs (pop1)
Compute the total power of all empires
i=1
Select the i (th) empire
Assimilate Colonies (pop2)
Information sharing between some colonies (pop3)
Revolve some colonies (pop4)
Improve imperialist (pop5)
pop1= Merge pop1, pop2, pop3, pop4 and pop5 for each empire
The members of front one add to archive
Updated archive based on non dominated solution
Sort the pop1 based on non dominated sorting and crowding distance
Pop1=Select NC (i) countries from empire (i) based on rank
The imperialist and the member of front 1 of colonies sort based non dominated sorting and crowding distance and selected rank1 as imperialist
i=i+1
i<nImp
Yes
No
Compute the total power of all empires
Give a weakest colony from the weakest empire to a stronger empire (Imperialist Competition)
Is there an empire whit no colonies?
Yes
No
Eliminate empire
Is number of Archive member greater than nArchive?
Yes
Archive=Select nArchive member from Archive based on rank
Is Stop Criterion statisfied?
Yes
No
Pareto solutions = Archive
Finish

Fig. 16. MOICA flowchart.

are selected as basic test problems. These standard test problems contain the activities and the predecessor relations between the activities as the basic test problems, yet some new data are required according to the mathematical model which are produced and described as follows:

- The required resources are in 2, 3, and 4 types in large scale problems and in 2 or 3 types in small problems.
- The number of performing methods for each activity is 3 methods in large problems and 2 or 3 in small problems.
- The normal duration of each activity in each method (d^n_{im}) is randomly produced by the uniform distribution $U(1,10)$.
- The crashing duration of each activity in each method (d^e_{im}) is randomly produced by $d^e_{im} = \max(ceil(d^n_{im}/3), ceil(d^n_{im} * random\ \ number))$
- For each activity in each method the normal required resources of each type (r^n_{imk}) is randomly produced by uniform distribution.
- The crashing required resources of each type for each activity in each method (r^e_{imk}) is randomly produced by $r^e_{imk} = ceil(r^n_{imk} * (1 + (1 - random\ \ number/2)))$
- The normal cost of each resource for each activity in each method (c^n_{imk}) is randomly produced by uniform distribution.
- The crashing cost of each resource for each activity in each method (c^e_{imk}) is randomly produced by

$$c^e_{imk} = \max(ceil((r^n_{imk} * c^n_{imk} * d^n_{im})/(d^e_{im} * r^e_{imk})), ceil(c^n_{imk} * (1 + (1 - random\ \ number/2)))) c^e_{imk}(isnan(c^e_{imk})) = 0$$

- The normal quality of the activities is randomly produced by uniform distribution $U(0.7, 1)$
- The crashing quality of the activities is the random percentage of the obtained normal quality. Obviously, for each activity, the crashing quality cannot be greater than the normal quality.

A brief description of the test problems which are selected from MRCPSP data set at *PSPLIB* is presented in Table 2. Also, the file name of these problems at the *PSPLIB* is mentioned in Table 2.

Table 2
Description of the test problems.

Project number	Problem description (*PSPLIB*)	Number of activities	Size of problem	Number of resources	Number of modes	Number of subsets
1	n015_9.mm	13	Small	3	3	3
2	j14.mm	14	Small	2	3	3
3	j146_6.mm	15	Small	2	3	5
4	j1464_10.mm	15	Small	3	3	2
5	j141_8.mm	15	Small	3	3	4
6	j149_9.mm	15	Small	3	3	5
7	r410_2.txt	17	Small	2	2	3
8	m12_4.mm	17	Small	3	2	4
9	m12_6.mm	17	Small	3	3	5
10	j1810_1.mm	18	Small	2	3	4
11	j189_10	18	Small	2	3	4
12	j209_8.txt	20	Small	2	3	5
13	j20.mm	21	Small	2	3	5
14	j20.mm	21	Large	3	3	6
15	j302_2	30	Large	2	3	5
16	j3013_9	30	Large	2	3	5
17	j301_8.txt	30	Large	3	3	5
18	j309_6.txt	30	Large	2	3	7
19	j301_7.txt	30	Large	3	3	6
20	j301_9	30	Large	3	3	6
21	j3024_4.mm	31	Large	2	3	5
22	j3064_4.txt	31	Large	2	3	6
23	j301_10.mm	31	Large	2	3	8
24	j3064_4.txt	31	Large	4	3	5
25	j303_10.txt	31	Large	4	3	6

4.2. Comparison metrics

To evaluate the performance of the algorithms, three comparison metrics are applied as described in this section.

4.2.1. Quality Metric (QM)

First, the non-dominated solutions obtained by all of the algorithms should be integrated together. Then, the non-dominated solutions from these integrated solutions should be determined. The percentage of the non-dominated solutions belonging to each algorithm is calculated as the algorithm *QM* [30]. The higher the *QM*, the better the algorithm.

4.2.2. Mean Ideal Distance (MID)

By applying this metric, the closeness between the Pareto solutions and the ideal point $(f_1^{best}, f_2^{best}, f_3^{best})$ can be calculated according to Eq. (18)

$$MID = \frac{\sum_{i=1}^{n} \sqrt{(f_{1i} - f_1^{best})^2 + (f_{2i} - f_2^{best})^2 + (f_{3i} - f_3^{best})^2}}{n} \tag{18}$$

where f_1^{best}, f_2^{best}, and $f_3^{best}$are the ideal points according to each fitness function and n is the number of non-dominated solutions. The lower the *MID*, the better the algorithm.

4.2.3. The Rate of Achievement to the objectives Simultaneously (RAS)

First, the ideal point should be considered and then the *RAS* can be calculated by Eq. (19).

$$RAS = \frac{\sum_{i=1}^{n} \left(\begin{array}{l} (|f_{1i} - f_1{}^{best}|)/(f_{1,total}^{max} - f_{1,total}^{min}) + (|f_{2i} - f_2{}^{best}|)/(f_{2,total}^{max} - f_{2,total}^{min}) \\ + (|f_{3i} - f_3{}^{best}|)/(f_{3,total}^{max} - f_{3,total}^{min}) \end{array} \right)}{n} \tag{19}$$

where f_1^{best}, f_2^{best}, and f_3^{best} are the ideal point values according to the objective functions. f_{1i}, f_{2i}, and f_{3i} are the objective function values of the solution *i*. $f_{1,total}^{max}$ and $f_{1,total}^{min}$ are the maximum and minimum values of the first objective function among all of the solutions, respectively. $f_{2,total}^{max}$ and $f_{2,total}^{min}$ are the maximum and minimum values of the second objective function among all of the solutions and so on. n is the number of total Pareto solutions.

4.3. Tuning the parameters

It is obvious that the proper choice of parameters has a striking impact on the performance of the algorithms. In order to calibrate the parameters of the proposed algorithm, there are several ways of statistically designing the experiments. Among the alternative experimental investigations to statistically calibrate the algorithm, the most frequently used and exhaustive approach is a full factorial experiment. However, it becomes increasingly difficult to carry out investigation when the number of factors significantly increases. To overcome this shortcoming, Taguchi developed Fractional Factorial Experiments (FFEs) that reduce the number of experiments while still keep sufficient information. The orthogonal arrays are applied in the Taguchi method to investigate a large number of decision variables with a small number of experiments [31,32].

Taguchi classified the factors into two main groups: controllable and noise factors. Noise factors are factors which we have no direct control on them and so elimination of these factors is impractical and often impossible. Therefore, this method seeks to minimize the effect of noises and to determine the optimal level of the important controllable factors as its two main targets.

Also, the relative significance of individual factors in terms of their main effects on the objective function is recognized by Taguchi. Moreover, he defined a transformation of the repetition data to another value called as a measure of variation. The transformation is the signal-to-noise (*S*/*N*) ratio which describes why this kind of parameter design is named robust design [33].

Table 3
Parameters and their levels for large scale problems.

Factor	Symbol	Level 1	Level 2	Level 3
P_c	A	0.6	0.7	0.8
ξ	B	0.15	0.25	0.35
β	C	0.2	0.35	0.5
MaxNe	D	6	7	8
npop	E	50	75	100
P_r	F	0.15	0.25	0.35
MaxDc	G	120	140	160
MinNe	H	1	2	3
narchive	J	50	60	70
nImp	K	6	8	10

Table 4
The orthogonal array L27.

Experiment number	A	B	C	D	E	F	G	H	J	K
1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	2	2	2	2	2	2
3	1	1	1	1	3	3	3	3	3	3
4	1	2	2	2	1	1	1	2	2	2
5	1	2	2	2	2	2	2	3	3	3
6	1	2	2	2	3	3	3	1	1	1
7	1	3	3	3	1	1	1	3	3	3
8	1	3	3	3	2	2	2	1	1	1
9	1	3	3	3	3	3	3	2	2	2
10	2	1	2	3	1	2	3	1	2	3
11	2	1	2	3	2	3	1	2	3	1
12	2	1	2	3	3	1	2	3	1	2
13	2	2	3	1	1	2	3	2	3	1
14	2	2	3	1	2	3	1	3	1	2
15	2	2	3	1	3	1	2	1	2	3
16	2	3	1	2	1	2	3	3	1	2
17	2	3	1	2	2	3	1	1	2	3
18	2	3	1	2	3	1	2	2	3	1
19	3	1	3	2	1	3	2	1	3	2
20	3	1	3	2	2	1	3	2	1	3
21	3	1	3	2	3	2	1	3	2	1
22	3	2	1	3	1	3	2	2	1	3
23	3	2	1	3	2	1	3	3	2	1
24	3	2	1	3	3	2	1	1	3	2
25	3	3	2	1	1	3	2	3	2	1
26	3	3	2	1	2	1	3	1	3	2
27	3	3	2	1	3	2	1	2	1	3

Table 5
$\overline{RDI}$s and weighted average of $\overline{RDI}$s of metrics for each experiment.

Experiment number	$\overline{RDI}$ (*Q*)	$\overline{RDI}$ (*MID*)	$\overline{RDI}$ (*RAS*)	D_i
1	0.7516	0.8055	0.8077	0.7736
2	0.2543	0.3112	0.3044	0.2757
3	0.3374	0.3922	0.3196	0.3448
4	0.9861	0.8918	0.8589	0.9418
5	0.2813	0.3592	0.5159	0.3438
6	0.4378	0.4764	0.4422	0.4464
7	0.9045	0.8518	0.8242	0.8779
8	0.4436	0.4873	0.4494	0.4535
9	0.3925	0.4889	0.5471	0.4427
10	0.5773	0.6442	0.6224	0.5997
11	0.6036	0.5098	0.3504	0.5342
12	0.3911	0.4476	0.4681	0.4178
13	0.6642	0.6279	0.642	0.6525
14	0.6114	0.5427	0.4806	0.5715
15	0.2136	0.2842	0.2325	0.2315
16	0.7093	0.7715	0.7631	0.7325
17	0.5079	0.6217	0.5941	0.5479
18	0.4675	0.4059	0.3916	0.4400
19	0.9943	0.9413	0.9848	0.9818
20	0.4924	0.5867	0.6331	0.5394
21	0.2473	0.1992	0.1964	0.2275
22	0.7763	0.8799	0.8717	0.8161
23	0.7082	0.6443	0.6431	0.6824
24	0.1279	0.1614	0.2314	0.1553
25	0.7952	0.8436	0.8298	0.8118
26	0.4431	0.5343	0.5339	0.4795
27	0.0031	0.0012	0.001	0.0023

Here, the term “signal” denotes the desirable value (mean response variable) and “noise” indicates the undesirable value (standard deviation). The aim is to maximize the signal-to-noise ratio. In the Taguchi approach, the objective functions are categorized into three groups of “the smaller-the better”, “the larger-the better”, and “the nominal-is-best”; for each of which a particular

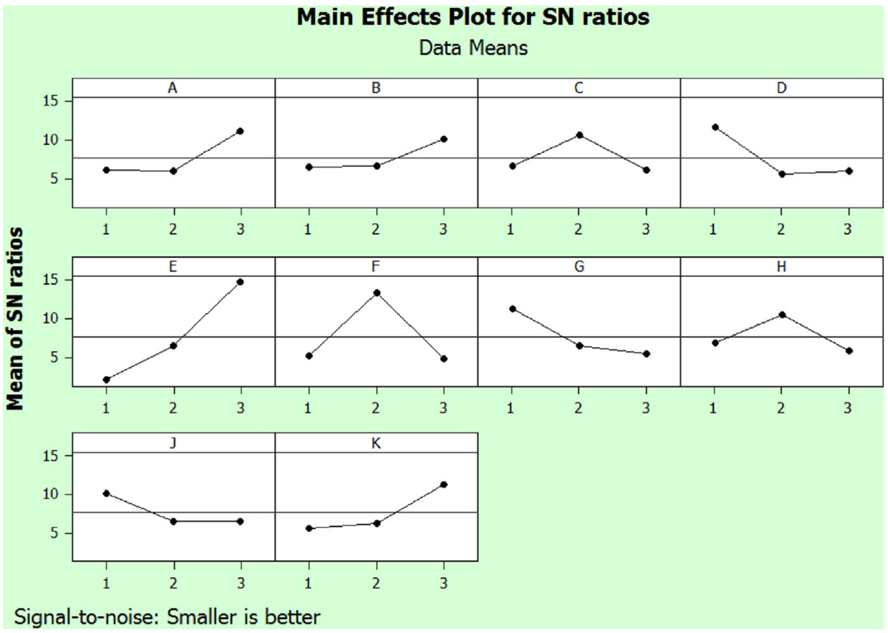

Fig. 17. The mean S/N ratio plot for each level of the factors in Taguchi methodology.

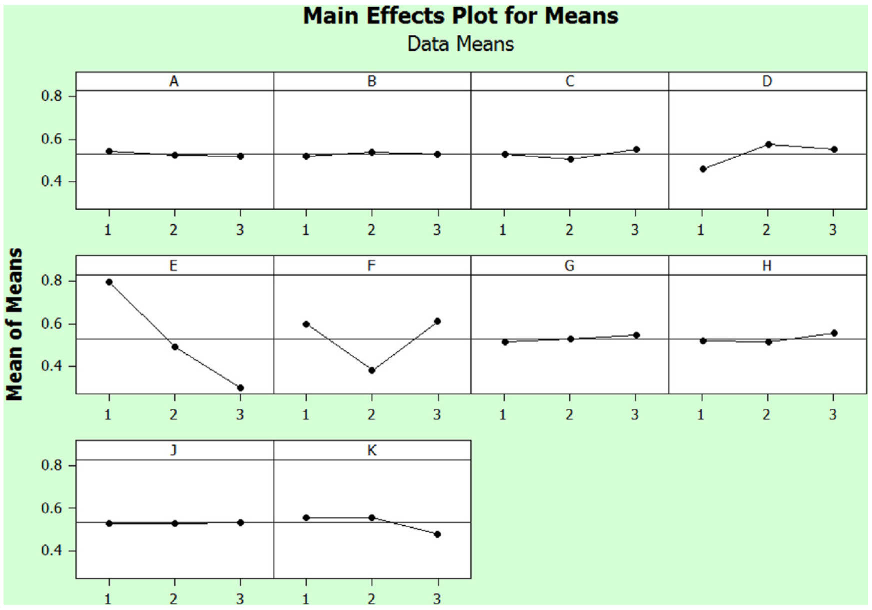

Fig. 18. The mean $\overline{RDI}$ plot for each level of the factors in Taguchi methodology.

Table 6
S/N ratio values.

Response table for signal to noise ratios
Smaller is better

Level	A	B	C	D	E	F	G	H	J	K
1	6.061	6.502	6.555	11.658	2.052	5.152	11.248	6.800	10.128	5.598
2	5.978	6.593	10.583	5.582	6.437	13.334	6.499	10.554	6.573	6.295
3	11.180	10.124	6.082	5.979	14.730	4.733	5.473	5.866	6.518	11.326
Delta	5.202	3.621	4.501	6.075	12.678	8.601	5.775	4.688	3.610	5.729
Rank	6	9	8	3	1	2	4	7	10	5

formula is used to calculate the (*S/N*) ratio. Most of the objective functions in scheduling are smaller-the-better type which is calculated by Eq. (20).

$$S/N \text{ ratio} = -10 \log \frac{1}{n}\left[\sum_{i=1}^{n} objective\ function_i^2\right] \tag{20}$$

In this study, the 10 control factors needing to be tuned for our proposed algorithm are: *Pc*; ξ; β; *MaxNe*; *npop*; P_r; *MaxDc*; *MinNe*; *narchive*; and *nImp*. For each control factor three levels are considered. These controllable factors and their respective levels for large scale problems are shown in Table 3. In addition, the symbols which are standing for the factors are presented in Table 3.

The considered orthogonal array with 10 factors and three levels in Taguchi method is L_{27}. The orthogonal array L27 is presented in Table 4.

As we divide the test problems into small size and large size problems, the Taguchi method is applied for parameter tuning in both scales separately. In this part, the parameter tuning for large size problems are described in detail by representing the step by step results while the final results of the parameter tuning for the small scale problems are also mentioned. Five test problems are chosen randomly in each scale. Each problem is being tackled five times to yield more reliable result, so we obtain 25 results for each trial. The best result among the 5 times run of each problem is considered as the result of that problem. Three metrics *QM*, *MID*, and *RAS* are applied as performance measures for each Taguchi experiment results. For each experiment, the results of these performance measures should be transformed to relative deviation index (*RDI*) which is calculated in Eq. (21).

$$RDI = \left|\frac{Algorithm_{solution} - Best_{solution}}{Max_{solution} - Min_{solution}}\right| \tag{21}$$

where $Algorithm_{solution}$ is the obtained value for each experiment by each performance measure, $Best_{solution}$ is the best value between the obtained values, $Max_{solution}$ and $Min_{solution}$ are the maximum and minimum values of each performance measure, respectively.

Then, we obtain the average of the *RDIs* for the five problems in each Taguchi experiment shown by $\overline{RDI}$ that is represented in Table 5.

The weighted average of the $\overline{RDI}$s for experiment *i* is calculated by Eq. (22) where w_j is the weight of metric *j* according to its importance and *m* is the number of metrics; $j=1,2,\ldots,m$. In this study *m* is equal to three.

$$D_i = \frac{w_1 \times RDI_{i1} + w_2 \times RDI_{i2} + \ldots.. + w_m \times RDI_{im}}{w_1 + w_2 + \ldots. + w_m} \tag{22}$$

Due to the importance of the quality metric, its weight is considered three and the weights of the other two metrics: *MID* and *RAS* are considered one.

The results obtained by Taguchi experiments are transformed into *S/N* ratio. The *S/N* ratios for each level and the results for each level are represented in Figs. 17 and 18, respectively. Regarding Figs. 17 and 18, the optimal levels of the factors A, B, C, D, E, F, G, H, J, K are A(3), B(3), C(2), D(1), E(3), F(2), G(1), H(2), J(1), K(3), respectively.

Figs. 17 and 18 and also the delta value represented in Table 6 demonstrate that *npop* is the factor which has most influence on the MOICA. P_r, *MaxNe*, *MaxDc*, *nImp*, P_c, *MinNe*, β, ξ, and *narchive* are the other effective factors on MOICA.

Table 7
Parameter setting values.

Factor	Symbol	Optimal level for large scale problem	Optimal level for small scale problem
P_c	A	0.7	0.7
ξ	B	0.15	0.15
β	C	0.35	0.35
MaxNe	D	7	6
npop	E	100	75
P_r	F	0.35	0.35
MaxDc	G	120	100
MinNe	H	1	1
narchive	J	70	50
nImp	K	10	8

Table 8
Computational results of metrics for the algorithms.

Problem number	*QM*					*MID*					*RAS*				
	SPEA-II	PESA-clustering	PESA-gridbased	NSGAII	MOICA	SPEA-II	PESA-clustering	PESA-gridbased	NSGAII	MOICA	SPEA-II	PESA-clustering	PESA-gridbased	NSGAII	MOICA
1	0	0	0.2821	0.2949	0.4231	0.8228	0.8846	0.6584	0.7318	0.7241	1.3861	1.4118	1.0367	1.0982	1.1175
2	0	0	0.1860	0.2016	0.6124	0.9267	1.2029	0.7336	0.8217	0.7695	1.4961	1.8860	1.1122	1.2405	1.1609
3	0	0	0.1250	0.1019	0.7731	0.6155	0.9325	0.6360	0.7663	0.7416	1.0080	1.2976	0.9729	1.1311	1.0846
4	0.0104	0	0.1211	0.0830	0.7855	0.5152	0.5728	0.6273	0.7143	0.6546	0.8190	0.9679	0.9431	1.0635	0.9920
5	0	0	0	0.0698	0.9302	0.9178	0.9592	0.6606	0.7084	0.5639	1.4774	1.4593	1.0733	1.0716	0.8473
6	0	0	0.0238	0.1571	0.8190	1.0302	0.7513	0.7675	0.7673	0.7549	1.6878	1.1790	1.2037	1.1599	1.1350
7	0	0	0.1310	0.1485	0.7205	0.6226	0.8858	0.6355	0.6737	0.6536	0.9537	1.3690	0.9348	0.9889	0.9753
8	0	0	0.2453	0.2013	0.5535	0.8225	1.1132	0.7447	0.7705	0.7571	1.2380	1.7298	1.1136	1.1511	1.1242
9	0	0	0.1905	0.1619	0.6476	0.8434	0.9980	0.6126	0.6143	0.6887	1.3534	1.4651	0.9299	0.9225	1.000
10	0	0	0.5319	0.2340	0.2340	0.6490	0.5024	0.5409	0.5831	0.7485	1.0680	0.7931	0.8150	0.8839	1.1098
11	0	0	0.4000	0.3143	0.2857	0.8193	0.4895	0.5397	0.5727	0.6540	1.2880	0.8202	0.8238	0.8729	0.9731
12	0	0	0.1256	0.1070	0.7674	0.7551	1.1371	0.6280	0.6651	0.5988	1.1818	1.7144	0.9420	0.9710	0.8858
13	0	0	0.1395	0.0698	0.7907	0.4469	1.0133	0.6095	0.6313	0.6396	0.7624	1.4773	0.9108	0.9480	0.9488
14	0	0	0.1592	0.2930	0.5478	0.9446	0.9701	0.6683	0.6431	0.6828	1.5578	1.5742	1.0181	0.9591	1.0162
15	0	0	0.3611	0.0093	0.6296	0.6475	0.5628	0.4178	0.6500	0.6756	1.0957	0.9038	0.6543	0.9868	0.9876
16	0	0	0.0194	0.1602	0.8204	0.5178	0.9099	0.6374	0.5768	0.6547	0.8695	1.5680	0.9808	0.8475	0.9684
17	0	0	0.0741	0.1435	0.7824	0.6199	0.8379	0.6469	0.6744	0.6730	1.0626	1.3893	1.0003	1.0185	1.0073
18	0	0	0.2865	0.0730	0.6404	0.6958	0.8459	0.6456	0.8212	0.7266	1.1723	1.3683	1.0349	1.2183	1.1142
19	0.0078	0	0.1172	0.2344	0.6406	0.5221	0.5142	0.6521	0.6435	0.6999	0.8498	0.8235	0.9784	0.9210	1.0077
20	0	0	0.0507	0.0507	0.8986	0.7088	0.7123	0.5609	0.6151	0.6720	1.1605	1.0929	0.8818	0.9186	0.9682
21	0	0	0.4254	0.0373	0.5373	0.7350	0.7526	0.5748	0.7132	0.6916	1.2456	1.2453	0.8692	1.0758	1.0302
22	0.0061	0	0.0915	0.2683	0.6341	0.5783	1.0811	0.6631	0.6883	0.6965	0.9594	1.4665	1.0589	1.0428	1.0425
23	0	0	0.1453	0.1512	0.7035	0.5524	0.5833	0.5219	0.6222	0.6742	0.9036	0.9300	0.8101	0.9247	1.0005
24	0	0	0.0200	0.2200	0.7600	0.7074	0.6381	0.5525	0.5191	0.5996	1.1919	1.0691	0.8810	0.8177	0.9144
25	0	0	0.0114	0.0170	0.9716	0.9391	0.6686	0.6264	0.6803	0.6072	1.3326	1.1026	0.9980	0.9839	0.8894

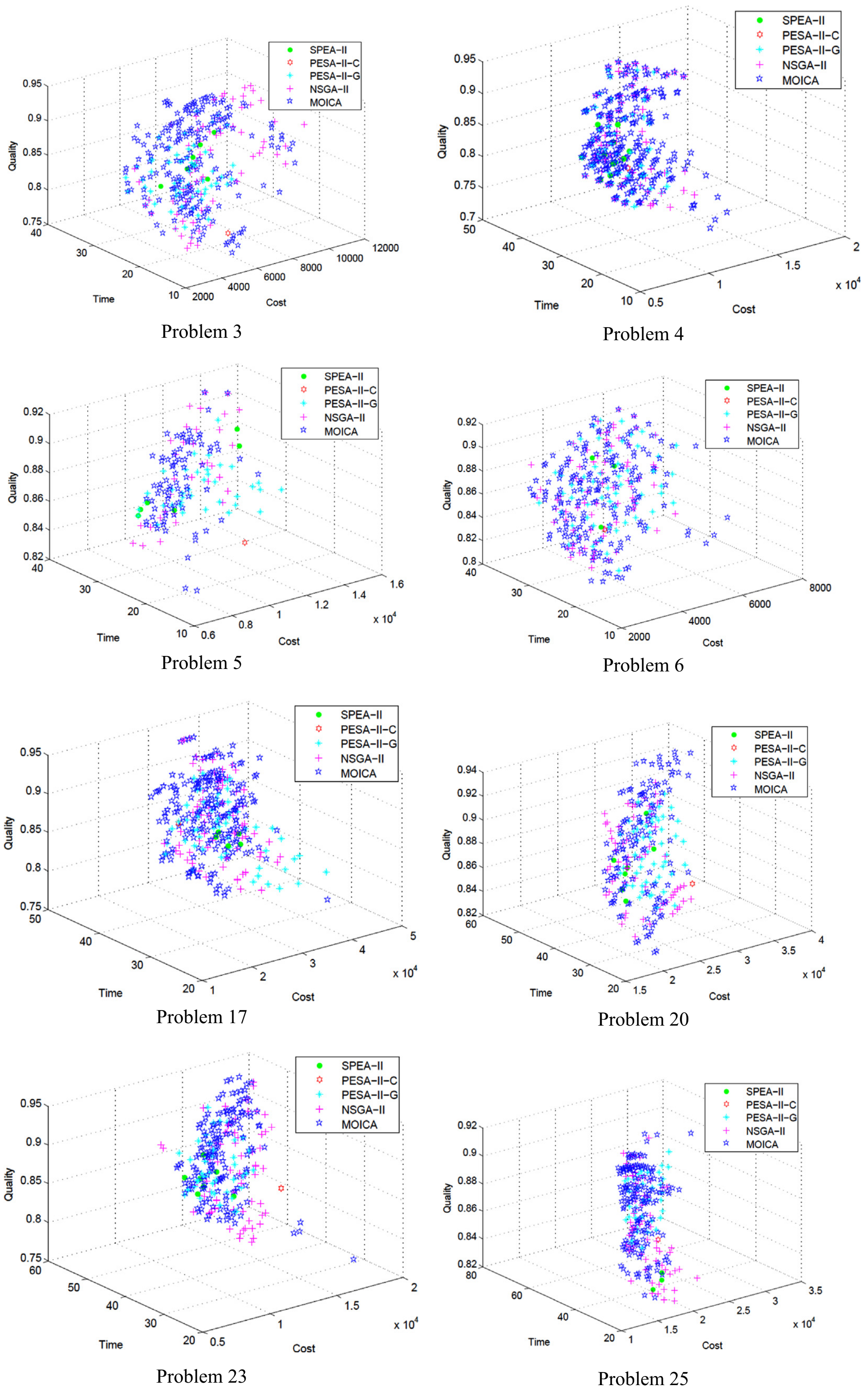

Fig. 19. Non-dominated solutions for each algorithm.

The optimal levels of the factors for the large scale problems are represented in Table 7. Also, the optimal levels of the factors for small scale problems are calculated and the result is shown in Table 7. The time consumed for solving of the problems is recorded for both large scale and small scale problems. Therefore, the other algorithms are run in a same time.

4.4. Computational outcomes

The performances of the five algorithms are represented in Table 8 based on the metrics values which are calculated for each algorithm in each of the 25 test problems.

Table 8 demonstrates that the MOICA optimal solutions are better in comparison with the other algorithms based on the metrics values. The obtained result of problem 5 reveals that the MOICA is the best algorithm based on the quality metric; $QM = 0.9302$ (the higher the QM the better the algorithm). MOICA is also the best one with respect

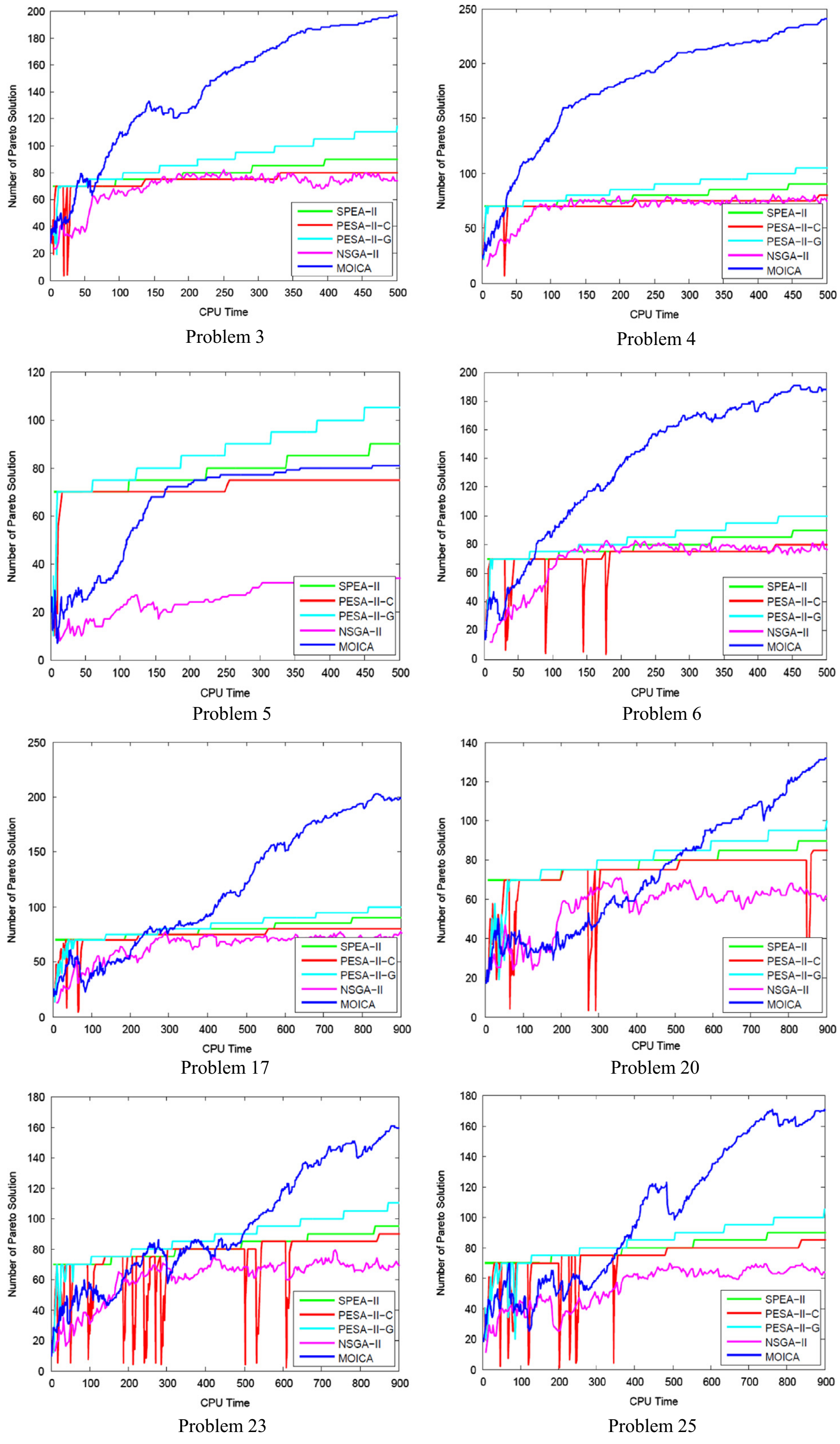

Fig. 20. Trend of non-dominated solutions for each algorithm.

to the mean ideal distance metric; *MID*=0.5639 (the lower is the better). In addition, it is obvious that MOICA has the lowest rate of achievement to the objectives simultaneously metric and so it is the best one; *RAS*=0.8473 (the lower is the better). Also, in the other test problems the MOICA has more appropriate results according to the metrics compared to the other algorithms.

Fig. 19 displays the Pareto solutions obtained from the algorithms in eight problems which are randomly selected from 25

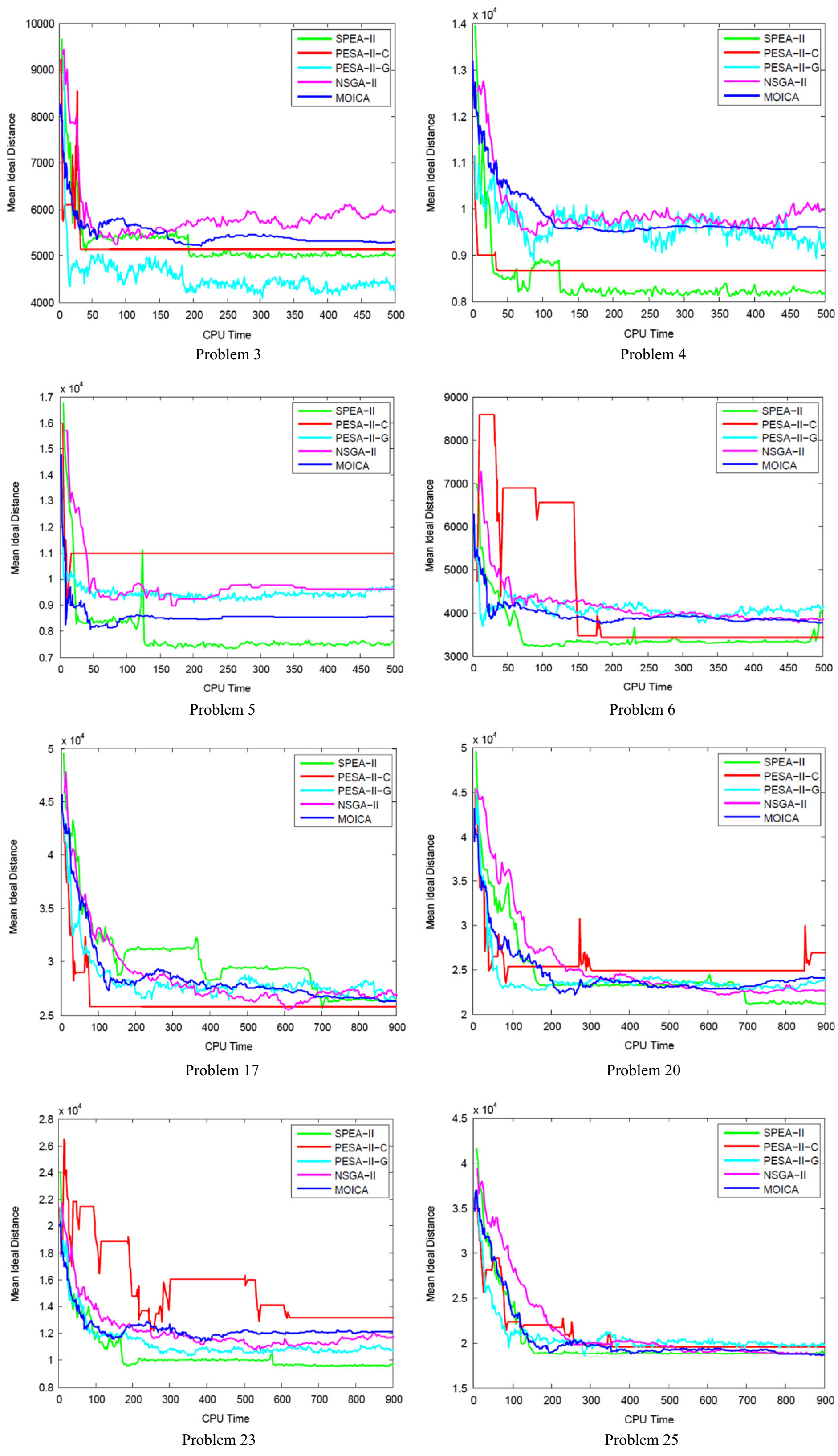

Fig. 21. Trend of *MID* for each algorithm.

test problems. As it is shown in Fig. 19, the Pareto solutions obtained by MOICA have higher quality compared to the other algorithms; in other words, the results of this algorithm can dominate most of the solutions obtained by the other algorithms.

Behavior of the algorithms in finding non-dominated solutions in an equal computational time is illustrated in Fig. 20 for the random eight problems. As it is obvious, the MOICA finds more non-dominated solutions in the iterations. Therefore, it is a more powerful algorithm in searching the non-dominated solutions in comparison with the other algorithms.

Also, the algorithms are examined in terms of *MID* for mentioned problems in equal computational time. With consideration to Fig. 21, it could be seen that the MOICA is a good algorithm according to the *MID* metric. It means that the ability of this algorithm to produce the non-dominated solutions with low distance from the ideal point is acceptable.

5. Conclusion and future research

A new mode-identity resource-constrained Discrete Time–Cost–Quality Trade-off Problem was introduced in this paper in which the goals were to minimize the project total time and cost while maximizing the project total quality. Also, the mode-identity problem was considered where the required resources for performing the activities were constrained. The problem was formulated mathematically. In order to solve this model, a Multi Objective Meta-heuristic Algorithm (MOICA) was presented. Twenty five test problems including large size and small size problems were solved with the proposed algorithm. The performance of this algorithm was evaluated by comparing it with four well-known multi objective algorithms: NSGAII, PESAII-clustering, PESAII-grid based, and SPEA2. The obtained results showed the effectiveness of this algorithm.

Some extensions of this research as a future study might be of interest. While in this paper, “minimum time”, “minimum cost” and “maximum quality” are considered as our problem goals, other objectives can also be considered. Other constraints can be added to this model to be as close as possible to the real world projects like the nonrenewable resources-constraint, generalized precedence relations, etc. The other extension of this research would be solving this problem by other meta-heuristic algorithms.

References

[1] Project management body of knowledge. PMBOK guide, 3rd ed.; 2008.
[2] Wuliang P, Chengen W. A multi-mode resource-constrained discrete time–cost tradeoff problem and its genetic algorithm based solution. Int J Proj Manag 2009:600–9.
[3] Salewski F, Schirmer A, Drexl A. Project scheduling under resource and mode identity constraints, model, complexity, methods, and application. Eur J Oper Res 1997;102:88–110.
[4] Hindelang TJ, Muth JF. A dynamic programming algorithm for decision CPM networks. Oper Res 1979;27(2):225–41.
[5] Liu SS, Wang CJ. Resource-constrained construction project scheduling model for profit maximization considering cash flow. Autom Constr 2008;17:966–74.
[6] Prabuddha DE, Dunne EJ, Ghosh JB. Complexity of the discrete time–cost tradeoff problem for project networks. Oper Res 1997;45(2):302–6.
[7] Vladimir GD, Gerhard JW. Hardness of approximation of the discrete time–cost tradeoff problem. Oper Res Lett 2001;29:207–10.
[8] Liu L, Burns S, Feng CW. Construction time–cost trade-off analysis using LP/IP hybrid method. J Constr Eng Manag 1995;121(4):446–54.
[9] Erenguc SS, Ahn T, Conway DG. The resource constrained project scheduling problem with multiple crashable modes: an exact solution method. Nav Res Logist 2001;48(2):107–27.
[10] Shtub A, Bard J, Globerson S. Project management: engineering, technology and implementations. Englewood Cliffs, NJ: Prentice Hall; 1994; 382–92.
[11] Butcher WS. Dynamic programming for project cost–time curve. J Constr Div 1967;93:59–73.
[12] Vanhoucke M, Debels D. The discrete time/cost trade-off problem: extensions and heuristic procedures. J Sched 2007;10:311–26.
[13] Boctor FF. Heuristics for scheduling projects with resource restrictions and several resource-duration modes. Int J Prod Res 1993;31:2547–58.
[14] Ahn T, Erenguc SS. The resource constrained project scheduling problem with multiple crashable modes: a heuristic procedure. Eur J Oper Res 1998;107: 250–9.
[15] Demeulemeester E, De Reyck B, Foubert B, Herroelen W, Vanhoucke M. New computational results on the discrete time/cost trade-off problem in project networks. J Oper Res Soc 1998;49:1153–63.
[16] Azaron A, Perkgoz C, Sakawa M. A genetic algorithm approach for the time–cost trade-off in PERT networks. Appl Math Comput 2005;168:1317–39.
[17] Yang T. Stochastic time–cost tradeoff analysis: a distribution-free approach with focus on correlation and stochastic dominance. Autom Constr 2011;43: 1–11.
[18] Chao-guang J, Zhou-shang J, Yan L, Yuan-min Z, Zhen-dong H. Research on the fully fuzzy time–cost trade-off based on genetic algorithms. J Marine Sci Appl 2005;4(3):18–23.
[19] Afshar A, Kaveh A, Shoghli OR. Multi-objective optimization of time–cost–quality using multi-colony ant algorithm. Asian J Civ Eng (Build Hous) 2007;8 (2):113–24.
[20] Shahsavari Pour N, Modarres M, Aryanejad MirB, Tavakoli Moghadam R. The discrete time–cost–quality trade-off problem using a novel hybrid genetic algorithm. Appl Math Sci 2010;42(4):2081–94.
[21] Zhang H, Xing F. Fuzzy-multi-objective particle swarm optimization for time–cost–quality tradeoff in construction. Autom Constr 2010;19:1067–75.
[22] Tareghian HR, Taheri SH. On the discrete time, cost and quality trade-off problem. Appl Math Comput 2006;181:1305–12.
[23] El-Rayes K, Kandil A. Time–cost–quality trade-off analysis for highway construction. J Constr Eng Manag 2005;131(4):477–86.
[24] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6(2):182–97.
[25] Corne DW, Knowles JD, Oates MJ. The Pareto envelope-based selection algorithm for multi-objective optimization. Lecture Notes Com Sci 2000;1917:839–48.
[26] Corne DW, Jerram NR, Knowles JD, Oates MJ, Martin J. PESA-II: region-based selection in evolutionary multi-objective optimization. In: Proceedings of the genetic and evolutionary computation conference (GECCO); 2001.
[27] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm; 2001.
[28] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialist competitive. In: Proceedings of the IEEE congress on evolutionary computation, Singapore; 2007.
[29] Kolisch R, Sprecher A. PSPLIB – a project scheduling problem library. Eur J Oper Res 1997;96(1):205–16.
[30] Moradi H, Zandieh M, Mahdavi I. Non-dominated ranked genetic algorithm for a multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem. Expert Syst Appl 2011;38(9):10812–21.
[31] Cochran WG, Cox GM. Experimental designs. 2nd ed.. USA: Wiley; 1992.
[32] Ross RJ. Taguchi techniques for quality engineering. New York: McGraw-Hill; 1989.
[33] Phadke MS. Quality engineering using robust design. USA: Prentice-Hall; 1989.
[34] Nabipoor Afruzi E, Roghanian E, Najafi AA, Mazinani M. A multi-mode resource-constrained discrete time–cost tradeoff problem solving using an adjusted fuzzy dominance genetic algorithm. Sci Iran, Trans E: Ind Eng 2013;20(3):931–44.