INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2021; 11(2):195-230

A MULTI-OBJECTIVE DECENTRALIZED MULTIPLE CONSTRUCTION PROJECTS SCHEDULING PROBLEM CONSIDERING PERIODIC SERVICES AND ORDERING POLICIES

M. Rostami[1*, †], M. Bagherpour[2] and Seyed M. H. Hosseini[1]
[1]*Department of Industrial Engineering and management, Shahrood University of Technology, Shahrood, Iran*
[2]*Department of Industrial Engineering, University of Science and Technology, Tehran, Iran*

ABSTRACT

In decentralized construction projects, costs are mostly related to investment, material, holding, logistics, and other minor costs for implementation. For this reason, simultaneous planning of these items and appropriate scheduling of activities can significantly reduce the total costs of the project undertaken. This paper investigates the decentralized multiple construction projects scheduling problem with the aim of minimizing 1) the completion time of the construction projects and 2) the costs of project implementation. Initially, a bi-objective integer programming model is proposed which can solve small-size problems using the $\varepsilon-\text{constraint}$ method. Then, a Priority Heuristic Algorithm (PHA), Non-dominate Sorting Artificial Bee Colony (NSABC) and Non-dominate Sorting Genetic Algorithm II (NSGA-II) are developed to handle large-size problems using a modified version of Parallel Schedule Generation Scheme (PSGS). The computational investigations significantly reveal the performance of the proposed heuristic methods over exact ones. Finally, the proposed methods are ranked using TOPSIS approach and metric definition. The results show that NSGA-II-100 (NSGA-II with 100 iterations), NSABC-100 (NSABC with 100 iterations) and PHA are ranked as the best known solution methods, respectively.

Keywords: decentralized multiple construction projects; periodic services; batch ordering; multi-objective; heuristic methods, artificial intelligence.

Received: 5 February 2021; Accepted: 15 May 2021

[*]Corresponding author: Department of Industrial Engineering and Management, Shahrood University of Technology
[†]E-mail address: rostami_m@shahroodut.ac.ir (Mohammad Rostami)

1. INTRODUCTION

Nowadays; due to the expansion of modern cities as well as the advancement of new technologies, construction projects are becoming more complex. The complexities behind the process of supplying resources and accomplishment of activities complicate the planning process of a project. In the decentralized construction projects, planning takes place in two main areas. The first area is related to the scheduling of the activities which special attention should be paid to the amount of available resources, precedence relationships, and the resource transfer time between activities. All of them can affect the completion time of the projects. The second area; is related to the costs incurred in order to implement a project. The most important costs are the constructing workshops costs (it is called resource pool in literature), the ordering costs of non-renewable resources, the cost of holding these resources, the costs of resource deterioration, as well as the costs of transferring resources between activities. In fact, these actions are taken in order to support the implementation of the activities undertaken. From planning point of view, failure to addressing these two areas at the same time may lead to increased costs, which ultimately increases the total costs. The most important costs that may be imposed due to the lack of decision integration are presented as follow:

- Increasing the time and cost of transferring resources between activities due to the lack of choosing the right location to build a resource pool.
- Increasing ordering costs due to improper planning of order quantity.
- Increasing holding or shortage costs due to improper planning of order time.
- Increasing deterioration costs due to improper planning of dispatching non-renewable resources to activities.

In all of the above situations, any kind of wrong decision might affect the total completion time of the construction projects. Therefore, an integrated decision on the planning of decentralized construction projects implementation is very important. In this paper, this important issue is being addressed.

The well-known Resource Constrained Project Scheduling Problem (RCPSP) and its many variants have been receiving an increasing attention from researchers. Table 1 shows the research carried out in this area in the recent years based on number of projects and corresponding objective functions.

Table 1 Segmentation of RCPSP literature

Objective/Project	Single project	Multi-projects
Single objective	(Davari and Demeulemeester, 2019); (Almeida et al., 2018);(Myszkowski et al., 2018); (Lacomme et al., 2017); (Bilolikar et al., 2016);(Kaveh et al., 2016); (He et al., 2016);(AlNasseri and Aulin, 2015) (Tran et al., 2015); (Myszkowski et al., 2015); (Van Peteghem and Vanhoucke, 2014); (Rostami et al., 2014)	(Chen et al., 2018); (Chakrabortty et al., 2017); (Beşikci et al., 2015); (Sonmez and Uysal, 2014); (Can and Ulusoy, 2014); (Beşikci et al., 2013)

Multi-objectives	(Tirkolaee et al., 2019);(El-Abbasy et al., 2016); (Laszczyk and Myszkowski, 2019);(Wang and Zheng, 2018);(Lu et al., 2018); (Gutjahr, 2015); (Capa and Ulusoy, 2015); (Tavana et al., 2014); (Gomes et al., 2014); (Ghoddousi et al., 2013)	(Geiger, 2017); (Chen et al., 2017); (Çebi and Otay, 2015); (Küçüksayacigil and Ulusoy, 2014); (Wang et al., 2014);

1.1 Literature review of decentralized multi-project scheduling problems

Unlike the centralized projects, the scheduling process of decentralized projects has been less widely considered in the literature. In decentralized construction projects, the resources transferring time between activities should be taken into account because they affect the project completion time. This problem was investigated by Yang and Sum (1993) for the first time. They considered equal resource transferring time between activities. In fact, these times are only between resource pools and activities, and no transferring time is considered between activities. By considering different resources transferring time between activities, Krüger and Scholl (2009, 2010) studied the same problem by developing a mathematical model. By the help of priority rules, they categorized projects according to their priorities and allocated constrained resources to projects. The great weakness of their heuristic algorithm was to prioritize project implementation at the beginning of the algorithm and the transfer of resources from higher priority projects to lower priorities not being possible. To address this weakness, Adhau et al. (2012) with using the multi agent system- One of the well-known methods in solving robotic problems - tried to obtain the appropriate solutions for decentralized multi-project scheduling problems. This decentralized algorithm uses auctions based negotiation (DMAS/ABN) approach that contributes to resolving resource conflicts and allocating multiple different types of shared resources amongst multiple competing projects. They divided the resources into two categories. There are a number of resources that can be locally available due to the abundance. Other category of resources is scarce and should be shared between projects as global resources. Resource pool location in project scheduling problems was considered by Rostami et al. (2017) for the first time. To solve decentralized multi-project scheduling problem, they used an artificial bee colony hybrid algorithm (HABC). They showed that considering the resources pool location simultaneously with the scheduling of activities reduces total costs. In an special case, Rostami and Bagherpour (2020) developed a lagrangian relaxation method to obtain lower bounds for this problem. Recently, Wang et al. (2019) developed a bi-objective optimization model to make the resource transfer decisions in a decentralized project scheduling problem. This model aims to minimize the transfer cost and maximize solution robustness in the presence of activity duration variability.

2.1 Literature review of project scheduling and material purchasing integration

The decision integration between material purchases and the project scheduling has been considered by some researchers from the past three decades. This problem was first investigated by Aquilano and Smith (1980), where a combination of critical paths method and material purchases was presented for the problem. After them, some researches were carried out on this subject. Among them Dodin and Elimam (2001) were considered this problem in situations where the durations are variable and awards are taken into consideration in purchasing. Also, Sajadieh et al. (2009) solved this problem with the aid of a genetic algorithm. Taking into account the batch ordering system, Fu (2014) present a hybrid genetic algorithm for the problem. Recently, Zoraghi et al. (2017) developed an integrated model for the planning of project implementation and material ordering simultaneously in a multi-mode resource constrained project scheduling problem under bonus–penalty policies.

3.1 Research gap

Based on the above mentioned discussion, only 7 research papers have been published on the decentralized projects scheduling problems so far, which only one article considered the resource pool location and periodic services altogether. Also, from researches conducted in the area of decision integration on project scheduling and material purchasing; only 2 papers were found in terms of incorporating batch ordering system in a centralized project environment. To the best of our knowledge, so far, no research was conducted on the consideration of batch order systems for the purchasing of materials in the decentralized construction projects environment; taking into account the resources pool location, periodic services, non-renewable resources deterioration, and sequence-dependent setup times. Thus; this paper deals with such a problem for the first time and introduce a bi-objective decentralized project scheduling problem as well. The first objective is to minimize the total project costs, and the second objective is to minimize the total completion time of the projects undertaken. Since the resource constrained multi-project scheduling problem is known as NP-hard (Salewski et al. (1997)), the problem developed in this paper by adding resources pool location will also be NP-hard. The rest of this paper is organized as follows: Section 2 defines the considered problem and presents a bi-objective mixed integer linear programming model. In order to solve this model with a commercial solver, a $\varepsilon-\text{constraint}$ method is developed in this section. Section 3 presents the solution methods for large-size instances, i.e. heuristic method, NSABC and NSGA-II that use a modified version of the PSGS. Section 4 provides computational results that include parameter tuning, the comparison of the results obtained by the proposed methods, and the ranking of the proposed methods based on defined metrics. Finally, section 5 summarizes the results and proposes future studies.

2. PROBLEM STATEMENT AND MODELING

In this section, first, the problem under consideration is comprehensively defined and then formulated by an integer programming model. The problem under study has the following assumptions and conditions:

- There is a program including several decentralized construction projects, meaning that project distances are noticeable. In fact, the projects are distinct from each other due to their noticeable distances

- The distances between activities are measured by the straight line method.
- The precedence relations between two consecutive activities are Finish to Start (FS).
- The problem includes one type of renewable and one type of non-renewable resource which are shown by indices of 1 and 2 respectively.
- The renewable resources have limitations and cannot be used by activities at any moment more than their capacity. These resources must be transferred between activities and have different setup times based on the sequence.
- The non-renewable resources due to local procurement are not limited, but should be purchased with the help of a batch order system. It is assumed that shortage is not allowed for these resources.
- To hold non-renewable resources, as well as to provide periodic services for renewable resources, we need to establish a resources pool for each resource type from among potential points. It is assumed that for each resource type only one resources pool should be established and it is possible to be established both resource pools in one potential point.
- Renewable resources should refer to the resource pool at specific times for periodic services and return to the activities after these services. Periodic service times are set at the beginning of the project planning, and the assumption is that at the time of the periodic service, if the resource is in implementation of an activity, the resource will continue the implementing of current activity and then refer to the resource pool.
- Non-renewable resources are stored after purchase in the resources pool. In the resource pool, the non-renewable resources do not deteriorate because of sufficient facilities in the pools. These types of resources begin to deteriorate after dispatching to activities, which cause the deterioration costs based on the linear time function.
- The first objective of the problem is to minimize the total cost of the construction projects, including the resource pool construction cost, the resources transferring cost, the holding and deterioration costs of non-renewable resources, as well as the ordering costs.
- The second objective of the problem is to minimize the project's total completion times. For defining mathematical model, Table 2 presents the parameters and decision variables.

Table 2: Parameters and decision variables

Section	Notation	Description
Sets	I	Set of Projects Activities
	$P_D(i)$	the direct predecessors of the ith activity
	$P_I(i)$	the direct/indirect predecessors of the ith activity
	L	the potential centers of the resource pool
	K	the required resources types
	T	the time horizon of the projects
Indexes	i, j	the activity number
	e	the dummy activity of total projects
	l	the centers of pool
	k	The type of resource
	t, u	the time

Parameters	D_{ij}	the time distance between *i* and *j*
	CT_{ij}	the transportation cost for each unit of renewable resource between *i* and *j*
	s_{ij}	the setup time of renewable resource when moves from *i* to *j*
	F_{lk}	the fixed cost of the building resource pool in the center *l* for type *k*
	r_{ik}	the resource type *k* needed for the *i*th activity
	R	the capacity of the renewable resources
	d_i	the duration time for the *i*th activity
	α_i	the deterioration coefficient for the *i*th activity
	η_t	Equal 1 if the service needs at time *t*
	CF_t	the fixed cost of each ordering in time *t*
	CB_t	the buying cost of each nonrenewable resource in time *t*
	CH	the holding cost of each nonrenewable resource at each unit time
	CD	the deterioration cost of each nonrenewable resource at each unit time
	M	A large positive number
Decision Variables	x_{ij}	The amount of renewable resource transfer from activity *i* to *j* ($x_{ij} = integer$)
	E_{lk}	Equals 1 if the resource pool related to resource *k* constructed in the center *l* ($E_l \in \{0,1\}$)
	z_{ij}	Equals 1 if the renewable resource transferred from *i* to *j* ($z_{ij} \in \{0,1\}$)
	y_{it}	Equals 1 if the processing of activity *i* starts at time *t* ($y_{it} \in \{0,1\}$)
	λ_{it}	Equals 1 if the processing of activity *i* starts before *t* ($\lambda_{it} \in \{0,1\}$)
	Q_t	The amount of nonrenewable resource ordered at time *t* ($Q_t = integer$)
	I_t	The inventory of nonrenewable resource in resource pool at time *t* ($I_t \geq 0$)
	γ_t	Equals 1 if a batch of nonrenewable resources is ordered at time *t* ($\gamma_t \in \{0,1\}$)
	θ_{it}	Equals 1 if the nonrenewable resource is sent to activity *i* at time *t* ($\theta_{it} \in \{0,1\}$)

In this part an integer linear programming model is presented which can obtain global optimal solution for the small-size instances. To simplify the solving procedure, a specific number has been assigned to each specific activity. Also, a dummy activity *e* is provided for the project completion where the duration, setup time, distances from all real activities and the non-renewable resources required is zero and the renewable resources required is *R*. Based on the defined parameters and decision variables, the linear mathematical programming model is defined as follows:

The objective function (1) minimizes the total cost of projects implementation. These costs include the resource pool construction cost, renewable resource transferring cost, the holding and deterioration cost of non-renewable resources, fixed costs of the ordering and variable costs of the resource purchasing. It should be noted that the fixed costs of ordering are not dependent on the order quantity, and in fact represents the logistics costs of each batch.

$$\begin{aligned} Min\ & \sum_{l}\sum_{k} F_{lk}E_{lk} \\ & + \sum_{i\in\{I\cup L\}}\sum_{j} x_{ij}CT_{ij} \\ & + CD\sum_{i} r_{i2}\alpha_i\left(\sum_{t} ty_{it} - \sum_{t} t\theta_{it}\right) \\ & + CH\sum_{t} I_t + \sum_{t} CF_t\gamma_t + \sum_{t} CB_tQ_t \end{aligned} \tag{1}$$

The objective function (2) minimize the completion time of projects that is equal to starting time of dummy activity.

$$Min\ \sum_{t} ty_{et} \tag{2}$$

Constraint (3) forces that one location must be selected as a resource pool for each resource type:

$$\sum_{l} E_{lk} = 1 \qquad \forall k \in K \tag{3}$$

Constraint (4) states that if the renewable resource pool is not established in the location *l*, this type of resource should not be getting out of this location:

$$z_{lj} \le E_{l1} \qquad \forall l \in L, j \in I \tag{4}$$

Constraint (5) states that it is not possible to transfer renewable resource from *i* to *j* more than the requirement of activity *i*. This constraint also prevents the transfer of resources from activity *i* to all activities that are its direct or indirect predecessors:

$$
\begin{aligned}
&x_{ij} \le z_{ij} r_{i1} \quad \forall i \in \{I \cup L\} \; (r_{l1} = R), j \notin P_I(i) \\
&x_{ij} = 0 \qquad \forall i \in I \; and \; j \in P_I(i)
\end{aligned}
\tag{5}
$$

Constraint (6) indicates that the total input resources to an activity should be equal to the sum of its output resources:

$$\sum_{j \in \{I \cup L\}} x_{ji} = \sum_j x_{ij} \qquad \forall i \in I \; and \; i \ne e \tag{6}$$

Constraint (7) forces that the total input renewable resources to an activity should be equal to the amount of requirement resources of that activity:

$$\sum_{j \in \{I \cup L\}} x_{ji} = r_{i1} \qquad \forall i \in I \tag{7}$$

Constraint (8) indicates that the total resources get out of the resource pool must be equal to the resource capacity:

$$\sum_l \sum_j x_{lj} = R \tag{8}$$

Constraint (9) states that the start time of the implementing of each activity should not be earlier than the completion of all corresponding predecessor activities:

$$\sum_t ty_{it} \ge \sum_t ty_{jt} + d_j \qquad \forall i \in I, j \in P_D(i) \tag{9}$$

Constraint (10) states that an activity can start to be implemented when the required renewable resource is completely received. The time of sweep among pool and activity must be added to the entry time, if the related resource needs time for periodic services (Part I). To calculate the sweep time, an auxiliary variable λ_{it} is used which can be calculated by Part II. It is also necessary to restrict the transfer of resources from an activity to activities that have an earlier start time that is controlled by constraint (Part III):

$$\begin{aligned}
&I)\ M\left(2-z_{ji}-E_{l1}\right)+\sum_t ty_{it} \geq \sum_t ty_{jt}+d_j+D_{ji}+s_{ji}+2D_{jl}\left(\sum_t \eta_t\left(\lambda_{jt}-\lambda_{it}\right)\right)\\
&\qquad\qquad\qquad\qquad \forall j\in\{I\cup L\}\,(d_l=0, y_{l0}=1), i\in I, l\in L\\
&II)\ \lambda_{it}=\sum_{u\leq t-1} y_{iu} \qquad \forall i\in I, t\in T \ \ and \ \ \lambda_{i1}=0\\
&III)\ z_{ij}\leq \sum_t ty_{jt}-\sum_t ty_{it} \qquad \forall i,j\in I
\end{aligned} \tag{10}$$

Constraint (11) states that an activity can be started to be processed where its required non-renewable resources receive from the resource pool:

$$\sum_t ty_{it} \geq \sum_t t\theta_{it}+\sum_l D_{li}E_{l2} \qquad \forall i\in I \tag{11}$$

Constraint (12) forces that each activity must only be start to process at one time:

$$\sum_t y_{it}=1 \qquad \forall i\in I \tag{12}$$

Constraint (13) forces that the non-renewable resources should only be dispatched to activity at one time:

$$\sum_t \theta_{it}=1 \qquad \forall i\in I \tag{13}$$

Equation (14) calculates the value of inventory at time *t*.

$$I_t=I_{t-1}+Q_t-\sum_i r_{i2}\theta_{it} \qquad (I_0=0) \quad \forall t\in T \tag{14}$$

Constraint (15) forces that the variable Q_t should not get a positive value if the resource is not ordered at time *t*:

$$Q_t\leq \gamma_t M \qquad \forall t\in T \tag{15}$$

Also, constraint (16) states that the sum of purchased non-renewable resources should not be less than the total required non-renewable resources:

$$\sum_t Q_t \geq \sum_i r_{i2} \tag{16}$$

Finally, equation (17) defines the model decision variables.

$$\begin{cases} E_{lk},\, z_{ij},\, y_{it},\, \lambda_{it},\, \theta_{it} \text{ and } \gamma_t \in \{0,1\} \\ x_{ij} \text{ and } Q_t \in Integer \\ I_t \geq 0 \end{cases} \tag{17}$$

As stated above, the proposed model is an integer linear programming model that can create optimal Pareto front for small-size instances. In this paper, the $\varepsilon-\text{constraint}$ method is used to conduct a single-objective model. In this method, the second objective function is considered as a constraint, i.e. $\sum_t ty_{et} < \varepsilon_{pf}$. By changing the value of ε_{pf} equal to *m* times in interval $[\varepsilon_{\min}, \varepsilon_{\max}]$, *m* Pareto solution can be generated. It should be noted that the quality of generated Pareto front depends on the value of ε_{pf} .

3. SOLUTION METHODOLOGIES

3.1 Modified parallel schedule generation scheme (MPSGS)

The priority rule-based scheduling methods are considered by researchers due to high speed in reaching to the final solution. Parallel schedule generation scheme is a time increment algorithm which, according to priority rules, schedules those that are most appropriate from eligible activities. The steps of PSGS algorithm are presented as follows: first a priority list of activities is provided as input. Then, by defining a variable that represents time, the eligible activities are scheduled based on priority. When the resource constraint is violated or the list of eligible activities is empty, the variable related to the time will be updated. This process continues until all activities are scheduled. This algorithm is primarily designed for the RCPSP problem and loses its effectiveness in the developed problems. For this reason, in this section, PSGS algorithm is modified based on the problem discussed in this article and is used to obtain feasible schedules in the proposed methods. It is inspired by the heuristic algorithm presented in Rostami et al. (2017) to develop MPSGS algorithm, in this paper. The variables and parameters used in this algorithm are defined as follows:

i, i' : index of activities,

j: index of an activity (or a pool) where the renewable resource exists in it,

$RSF(t)$: set of actual renewable resources at time *t*,

$RSG(t)$: set of potential renewable resources at time *t*,

$A(t)$: set of eligible activities at time *t*,

$B(\eta)$: set of processing activities at times of periodical services,

$MRS(i)$: table of descended sorting cumulative renewable resources for activity *i*,

st_i : start time of processing activity *i*,

$\omega_i(t)$: allocable renewable resources in activity *i* at time *t*,

Dis_{ji} : time distance of renewable resource in activity (or pool) *j* from activity *i*, Also, in this algorithm some inputs are required which includes the location of renewable ($l1$) and non-renewable ($l2$) resource pool, the prioritized list of activities (PA), the prioritized list of renewable resources allocation to activities (PRA_i), as well as the dispatching time of non-renewable resources to each activity (dt_i).

At the beginning, the variable related to the time (t) is considered equal to zero. In this step, the set $A(t)$ is formed that consists of activities whose all direct and indirect precedence activities have been terminated. Also, the sets of $RSF(t)$, $RSG(t)$ as well as variable $\omega_i(t)$ are calculated. Actual resources are those resources that are ready to be assigned to an activity at time *t*, i.e. resources that are either in the pool or in an activity that has been completed before *t*. The distance of actual resources located in activity (or pool) *j* from activity *i* can be calculated from equation (18):

$$Dis_{ji} = D_{ji} + s_{ji} \qquad \forall j \in RSF(t) \tag{18}$$

Unlike actual resources, a potential resource is a renewable resource that is running in an activity and released at a time after t. The distance of potential resources in activity *j* from activity *i* can be calculated by relation (19):

$$Dis_{ji} = st_j + d_j - t + D_{ji} + s_{ji} + 2D_{jl1}\left(\sum_t \eta_t\left(\lambda_{jt} - \lambda_{it}\right)\right) \qquad \forall j \in RSG(t) \tag{19}$$

Also, $\omega_i(t)$ is equal to the sum of actual and potential resources that are at time *t* in activity *i* and are not assigned to another activity.

Then, based on the PA list, the activity with the highest priority is selected from $A(t)$. In the next step, the $MRS(i)$ table is formed for the activity with the help of the PRA_i. Initially, the activities and resource pool are sorted according to the PRA_i list and are placed in the first column of $MRS(i)$ table. In the second column, the cumulative $\omega_i(t)$ are computed for these activities. Then r_{i1} value is compared with the cumulative values of the second column. The first row in which the relation $r_i \le \sum \omega_j(t)$ is true indicates that the activity *i* for its procedure needs those resources placed from the first up to that specified rows. After the formation of $MRS(i)$, the starting time of activity *i* is calculated from equation (20):

$$st_i = \max\left\{\max_j\left\{Dis_{ji}\right\}, dt_i + D_{l2i}\right\} \tag{20}$$

In Eq. (20), *j* represents the activities that the renewable resources allocated to the activity *i*. Also, D_{l2i} represents the distance of non-renewable resource pool from activity i.

Then the scheduled activity is removed from $A(t)$ and $RSF(t)$, $RSG(t)$ and $\omega_i(t)$ are updated. When $A(t)=\varnothing$ or $RSF(t)=\varnothing$ the value of *t* will be updated. In this way, *t* increases to a point which at least one potential resource turns into an actual resource. This amount increase is displayed by t_step. If during the implementation of the activity or at the end of implementation the due date of period servicing is reached, the resources must return to the pool and after servicing come back again to the place of activity (21):

$$t\_step = min\left\{ st_j + d_j - t + 2D_{l1j} \sum_{u=t}^{st_j+d_j} \eta_u \,\middle|\, j \in RSG(t) \right\} \tag{21}$$

And this process continues until all activities are scheduled. The pseudo- code of MPSGS algorithm is described as follows:

```
Inputs:
    Location of renewable resources
    Location of nonrenewable resource
    PA list
    PRA_i list
    dt_i values
Resources allocation and scheduling:
Until scheduling all of activities do
    Step0. Set t=0,
    Step1. Update $A(t)$, $RSF(t)$, $RSG(t)$ and $\omega_i(t)$,
    Step2. Choose the most priority activity from $A(t)$,
    Step3. Construct the $MRS(i)$ for this activity based on PRA_i list,
    Step4. Calculate the starting time of this activity
    Step5. Remove the scheduled activity from $A(t)$,
    Step6. If $RSF(t)=\varnothing$ or $A(t)=\varnothing$ then go to Step 7 otherwise go to Step2,
    Step7. Update t and go to Step1,
End.
```

3.2 Priority rule-based heuristic algorithm (PHA)

In this section, a two-stage priority rule-based heuristic algorithm is proposed to generate a local optimum Pareto front in a short time. This algorithm, according to the definition of different priority rules, can create a Pareto front with up to 64 different solutions. At the first stage, allocation of resources and scheduling of activities are specified while planning of non-renewable resources order and also dispatching them to the activities are considered as the second stage.

As stated in Section 3.1, the MPSGS method has 5 inputs. Here, for each input, different priority rules are given that the combination of these rules creates 64 different settings ($8\times2\times2\times2$). In order to generate diverse solutions, it has been tried to use priority rules, some of which will minimize the completion time of projects, and others to minimize imposed costs.

There are two types of cost-based and time-based point of views for renewable resource pool location. From the location point of view, it can also be based on P-median or P-center problems. Finally, it is also necessary to consider the activities that are being processed during the periodic services. For this reason, in order to select the optimal location for the renewable resource pool, 8 priority rules can be presented, as shown in Table 3.

Table 3: Priority rules related to location of renewable resources

Priority rule	Point of view 1	Point of view 2	Point of view 3	Calculation method
1	Without considering the activities in periodic service times	Time-oriented	P-median	$Arg \min_{l \in L} \left\{ \sum_{i \in A(0)} r_{i1} \left(D_{l1i} + s_{l1i} \right) \right\}$
2			P-center	$Arg \min_{l \in L} \left\{ \max_{i \in A(0)} \left\{ r_{i1} \left(D_{l1i} + s_{l1i} \right) \right\} \right\}$
3		Cost-oriented	P-median	$Arg \min_{l \in L} \left\{ F_{l1} + \sum_{i \in A(0)} r_{i1} CT_{l1,i} \right\}$
4			P-center	$Arg \min_{l \in L} \left\{ F_{l1} + \max_{i \in A(0)} \left\{ r_{i1} CT_{l1,i} \right\} \right\}$
5	With considering the activities in periodic service times	Time-oriented	P-median	$Arg \min_{l \in L} \left\{ \sum_{j \in A(0)} r_{i1} \left(D_{l1i} + s_{l1i} \right) + \sum_{j \in \beta(\eta)} 2 r_{j1} D_{l1,j} \right\}$
6			P-center	$Arg \min_{l \in L} \left\{ \max \left\{ \max_{i \in A(0)} \left\{ r_{i1} \left(D_{l1i} + s_{l1i} \right) \right\}, \max_{j \in \beta(\eta)} \left\{ 2 r_{j1} D_{l1,j} \right\} \right\} \right\}$
7		Cost-oriented	P-median	$Arg \min_{l \in L} \left\{ F_{l1} + \sum_{i \in A(0)} r_{i1} CT_{l1,i} + \sum_{j \in \beta(\eta)} 2 r_{j1} CT_{l1,j} \right\}$
8			P-center	$Arg \min_{l \in L} \left\{ F_{l1} + \max \left\{ \max_{i \in A(0)} \left\{ r_{i1} CT_{l1,i} \right\}, \max_{j \in \beta(\eta)} \left\{ 2 r_{j1} CT_{l1,j} \right\} \right\} \right\}$

Unlike renewable resources, the non-renewable resources have no considerable periodic services, so they do not need to return back to the pool. Therefore, from the location point of view, the non-renewable resource pool location is similar to the P-median problem. Table 4 shows the priority rules for choosing the location of non-renewable resource pool.

Table 4: Priority rules related to location of nonrenewable resources

Priority rule	Point of view 1	Point of view 2	Calculation procedure
1	Time-oriented	P-median	$Arg \min_{l \in L} \left\{ \sum_{i} r_{i2} D_{l2,i} \right\}$

2	Cost-oriented	P-median	$Arg \min_{l \in L} \left\{ F_{l2} + \sum_i r_{i2} CT_{l2,i} \right\}$

The priority list of activities is also created through two predefined priority rules, the earliest start time (EST) and the shortest processing time (SPT). In order to produce a priority list of allocation of renewable resources to each activity, two different priority rules are used. In the first rule, priority is given based on the lowest value Dis_{ji}. In the second rule, priority is given to the highest value $\omega_j(t)$.

Assuming there is no limit to the availability of non-renewable resources at a proper time, for each combination created from the above priority rules, the start time of the activities is determined. After determining the starting time of the activities, the purchasing time of non-renewable resources must be determined, as well as dispatching them to the activities. The first condition to be taken into account is that the time of dispatching the non-renewable resources to activity *i* should not be longer than $max\_dt_i = st_i - D_{l2i}$. In this equation, *l2* is the location of non-renewable resource pool, and max_dt_i is the latest allowed time to dispatch the non-renewable resource to activity *i*. In order to determine the time of dispatching the resources to each activity *i*, the holding cost of resources at the pool and the deterioration cost of the resources at the site of activity should be first determined. It should be determined which cost is lower. To do this, there should be a simple comparison between the cost coefficients for each activity. For each activity *i* that $\alpha_i CD \geq CH$ is true, it is better to hold the nonrenewable resources in the pool until max_dt_i and then be dispatched to the activity site. The dispatched resources are used immediately after arrival. Also, for each activity *i* that $\alpha_i CD \geq CH$ is not true, it is better to dispatch the resources to the activity immediately after procurement and remain in activity site until processing begins.

A greedy search algorithm is also used to plan the purchase of nonrenewable resources. In this way, at the beginning of the algorithm, it is assumed that for each activity *i* the required resources are ordered once and the ordering time is equal to max_dt_i. The purchasing plus deterioration cost of this feasible solution is shown at the beginning of the algorithm with FF_0 and is equal to Eq. (22):

$$FF_0 = \sum_i CF_{max\_dt_i} + \sum_i r_{i2} CB_{max\_dt_i} + CD \sum_i r_{i2} \alpha_i D_{l2i} \qquad (22)$$

Then activities are sorted ascending by max_dt_i.

Using the greedy search algorithm, the first activity is compared with the second activity, whether it is cost-effective to order the required resource for the second activity simultaneously with the resources related to the first activity. For this purpose, the order cost is calculated using two modes, i.e. simultaneous order and separate order. If the cost of the simultaneous order be lower, then it is economical to order the required resource for both these activities simultaneously. Then the third activity is compared with these two activities, and this process continues. If the cost of separate order be lower, the first activity is ordered individually and removed from the list of activities and the second activity is considered as the first activity and the next activity takes place in the second activity position. This comparison continues until the last activity. The amount of cost after applying the changes can be calculated from equation (23), in which *[i]* represents the activity in rank *i*, and *g* is the index of the step of calculation.

$$\begin{aligned} FF_g &= FF_{g-1} + r_{[2],2}\left(CB_{max\_dt_{[1]}} - CB_{max\_dt_{[2]}}\right) - CF_{max\_dt_{[2]}} \\ &+ min\left\{\alpha_{[2]}CD, CH\right\} \times \left(max\_dt_{[2]} - max\_dt_{[1]}\right) \times r_{[2],2} \end{aligned} \tag{23}$$

After determining optimal ordering and the scheduling of the purchase orders for non-renewable resources, the two objectives function of the problem can be calculated. To save the optimal Pareto's results, the Pareto set is used. This set will update every time a solution is obtained, so that all the solutions in it are non-dominant. After executing the algorithm for all 64 different settings, the solutions in the Pareto set are identified as the output of the algorithm. The pseudo-code of PHA is as follows:

```
For setting 1 to 64 do
  Stage1. Location/Resource allocation/Scheduling:
        Determine location of renewable resources
        Determine location of nonrenewable resources
        Determine PA list
        Determine $PRA_i$ list
        Set $dt_i$=max_ $dt_i$
        Schedule activities with MPSGS
  Stage2. Resources ordering and dispatching:
      Run greedy search algorithm,
          Determine optimum resource ordering and dispatching,
  Calculate objective1 and objective2,
Update Pareto set,
End for,
Output optimum Pareto set.
```

3.3 Non-dominated sorting artificial bee colony (NSABC)

The basic artificial bee colony method is one of the population-based meta-heuristic methods inspired by the intelligent behavior of bee honey in finding food. This algorithm was first developed by Karaboga and Basturk (2007). Subsequently, various versions of this algorithm were proposed for solving multi-objective problems. Among them can be noted the study of Akbari et al. (2012) in which a MOABC algorithm was developed with the aid of a grid-based approach to generate pareto front solutions. Akay (2013) presented three different multi-objective methods by changing the selection and fitness strategies in the original ABC. Also, Zhong et al.(2014), by dividing the original colony into three sub-colonies, presented a method called dMOABC. But recently, a new development of the ABC algorithm for solving multi-objective problems has been proposed by Kishor et al.(2016), entitled NSABC. They used the non-dominated sorting and crowding distance used in the NSGA-II algorithm (Deb et al. (2002)) to obtain optimal and diverse solutions. In this paper, we will adapt their approach to the problem and use it to solve large-size instances.

The NSABC algorithm is structurally similar to the original ABC algorithm, and uses only one archive to store the optimal Pareto front. The pseudo-code of the NSABC algorithm is as follows, as proposed by Kishor et al. (2016):

```
Step1. Population initialization
Step2. Generate archive
Step3. Iteration=1
    Step4. While Iteration<Iter_max do
    Step5. Employee bee phase
    Step6. Onlooker bee phase
    Step7. Scout bee phase
    Step8. Update archive
    Step9. Iteration=Iteration+1
    Step10. End while
Step11. Return archive
```

In the structure of this algorithm, as the original ABC algorithm, the food sources number is selected randomly, that is the half of the population size. In each food source, an employee bee is looking for suitable food and nectar extraction and returns to the hive and begins to dance in the dance area. In this area, onlooker bees will select the most appropriate nectars with a probability function and update the archive through it. If a food source, after several search stages, namely *Max_trial*, has no proper nectar, is abandoned by the employee bee and a scout bee is used to search for a new food source around the hive. This process will continue until the stop condition.

In this algorithm, the archive is updated with the help of the non-dominated sorting and crowding distance used in the NSGA-II algorithm. In updating the archive, the most important parameters for selecting are the rank of the solution and the crowding distance, respectively. A solution with a lower ranking and greater crowding distance is more favorable for selecting. The members of population are placed in different fronts. Each solution is assigned a rank such that all non-dominated solutions are assigned a rank 1; the second best solutions which are only dominated by rank 1 are assigned rank 2; and so on. The crowding distance is calculated for each member of each rank. This parameter indicates the degree of proximity of the member to other members of the rank. The larger of this parameter will lead to diversity in the population. The crowding distance of solution *i* and related to objective function *k* is equal to:

$$d_i^k = \frac{\left| f_{i+1}^k - f_{i-1}^k \right|}{f_{\max}^k - f_{\min}^k} \tag{24}$$

So that f_{i-1}^k and f_{i+1}^k are the values of objective function *k* for the sorted neighbors before and after of *i*. Also, $f_{\max}^k$ and $f_{\min}^k$ are maximum and minimum of objective function *k*, respectively. Finally, the crowding distance of solution *i* will be equal to:

$$CD_i = \sum_{k=1}^{n} d_i^k \tag{25}$$

To design a quick method, the objective functions are obtained with the help of the MPSGS structure presented in Section 3.1. Given that this algorithm has 5 inputs, hence the form of the genotype will be as follows:

A matrix $2 \times l$ is considered to determine the locations for the construction of the renewable and non-renewable resource pool (the first row refers to the renewable resource pool). The numbers inside this matrix are in range [0,1]. In each row, the largest number is selected as the location of the corresponding pool. Fig. 1 shows a representation of resource pools location.

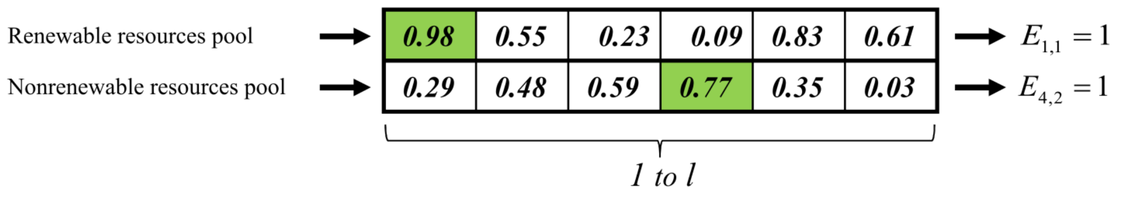

Figure 1. Representation of resources pools location

In order to generate the *PA* list, a string with n (the number of activities of the whole projects) cells is used that contains numbers in the interval [0,1]. The larger the number of a cell represents the higher the priority for processing the activity corresponding to the cell’s index. Fig. 2 shows the representation of the *PA* list.

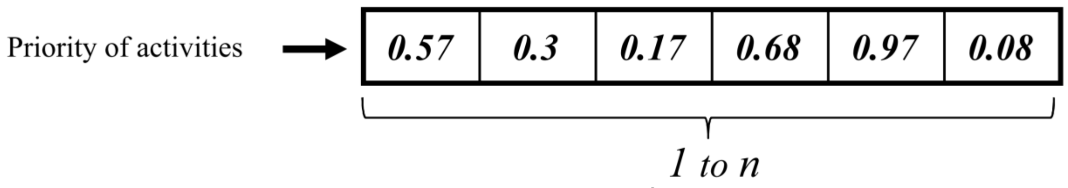

Figure 2. Representation of *PA* list

Also, for each activity *i*, a string with n + 1 cells and numbers in the interval [0,1] is used to generate the PRA_i list. The cells from 1 to *n* are related to the activity index, and the last cell in the string is related to the location of the renewable resource pool. The larger numbers reveal higher priorities for transferring resources from the activity corresponding to the cell's index to activity *i*. In order to avoid generating infeasible solutions, it is forbidden to transfer resources from activities that the activity *i* is predecessor of them to *i*. Fig. 3 shows the representation of PRA_i list for each activity *i*. In this figure it is supposed that activity *i* is a predecessor of activities 4 and 6.

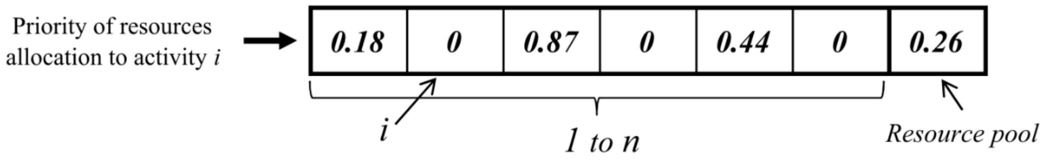

Figure 3. Representation of PRA_i list

Ultimately, a string with *n* cells is used to determine the times of dispatching non-renewable resources to activities. The numbers assigned to this string are integer in the interval [0, T], where T is the time horizon of the projects. Fig. 4 illustrates the representation of determining dt_i.

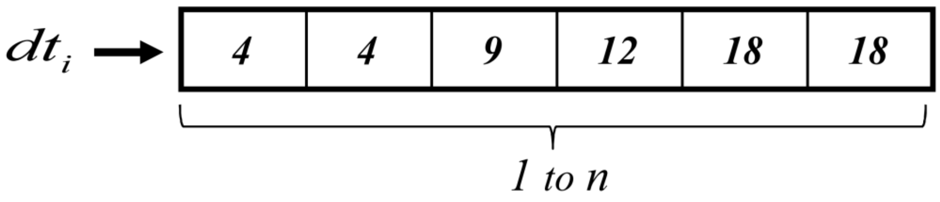

Figure 4. Representation of dt_i setting

It should be noted that the feasibility of generated solutions must be checked out continuously at any iteration, because during generation of any new solution, the created strings may violate the constraints of the problem. Here, the only string that may be infeasible is the *PA* list, because there are precedence relations between activities. In this string, if the priority of activity *j* exceeds at least one of its predecessors, the priority number of activity *j* will be replaced by its predecessors. This process continues until generating a feasible string.

Initially, the random food sources are generated as much as population size, i,e. NP. Half of this population is allocated to the employee bees and half to the onlooker bees. Equation (26) is used to generate any random food source *i*:

$$X_i(d) = X_{\min}(d) + rand_i(0,1)\left(X_{\max}(d) - X_{\min}(d)\right) \qquad \forall d \in \{1,...,D\}, i \in \{1,...,NP\} \tag{26}$$

In this equation, $x_{\min}(d)$ and $x_{\max}(d)$ are respectively the lower and upper bound for each dimension *d* and $d \in \{PA, PRA_i, dt_{ij}\}$. Then the objective function of each food source is calculated. Also for each food source, the parameter of trial is considered equal to zero. At the beginning of the algorithm, all generated random food source is stored in the archive.

In the employee bee phase, each employee bee tends to select an employee bee in its neighborhood ($1\,to\,NP/2$) randomly (with the index *m*). Also, a dimension is chosen randomly from among the dimensions (with the index *j*). Then, each food source is optimized with the aid of equation (27):

$$V_i(j) = X_i(j) + \omega_1 \times r_i(j) \times \left(X_i(j) - X_m(j)\right) \tag{27}$$

In this equation, ω_1 controls the generation of the solution in the neighborhood *i*, and $r_i(j)$ is a random number in the interval [-1,1]. After generating candidate solution V_i, if V_i dominates X_i, then the candidate solution replaces X_i and trial = 0. If X_i dominates V_i, then the replacement does not happen, and the trial increases by one unit. If V_i and X_i are non-dominated, then a random solution is taken from these two as the new position of the food source *i* and then the trial increases by one unit.

In the onlooker bee phase, firstly, the employee bees share information from their food sources in the dance area. Then each onlooker bee selects one of the food sources to update their position. The selection happens through a probability density function. In the selection process, the chance of choosing more fitness food source is higher. In this paper, the fitness function of each food source can be calculated according to Eq. (28), which is inspired by the research of Akbari et al. (2012):

$$fit(X_i) = \frac{dom(i)}{FoodNumber} \tag{28}$$

In this equation, $dom(i)$ shows the amount of food sources dominated by the food source *i*. With the help of the fitness function (28), a food source with probability P_i is selected by the onlooker bees with the aid of a roulette wheel method:

$$P_i = \frac{fit(X_i)}{\sum fit(X_i)} \tag{29}$$

After choosing the appropriate food sources by the onlooker bees, the candidate solutions are obtained by equation (30):

$$V_i(j) = X_i(j) + \omega_2 \times r_i(j) \times \left(X_i(j) - X_k(j) \right) \tag{30}$$

In this equation, k is related to the selected employee bee and the j index is chosen randomly from the dimensions. If V_i dominates X_i, then the candidate solution replaces X_i. If X_i dominates V_i, then the replacement does not happen. If V_i and X_i are non-dominated, then a random solution is taken from these two as the new position of the food source *i*. If a food source after several attempts (*Max_trial*) fails to generate suitable nectars, it is abandoned by the employee bee and a new food source is generated randomly with the help of the scout bees.

At any iteration, the archive should be therefore updated. All the solutions in the current swarm are merged with the archive solutions and form a new population. NP solutions are selected by using the update method mentioned above and thus the archive will be updated. The process of this algorithm continues to reach the termination condition, i.e. the maximum iteration.

3.4 Non-dominated sorting genetic algorithm II (NSGA-II)

Genetic algorithm is one of the best – known population-based evolutionary meta-heuristic methods, first proposed by Holland (1975). Many versions of this algorithm have been already developed for solving multi-objective problems. One of the most famous multi-objective genetic methods is called NSGA-II, which was developed by Deb et al (2002). Like GA, his method uses crossover and mutation operators to generate new offspring. But for ranking and selecting the suitable population for the next generation, it uses the non-dominated sorting and crowding distances described in Section 3.3.

In this paper, the representation of the method is similar to the NSABC algorithm presented in Section 3.3 and also the MPSGS structure presented in Section 3.1 is used to calculate the objective functions. Initially, NP chromosomes are generated randomly that their objective functions are computable. Then, P_c parent is selected from the initial population, which is done by the binary selection mechanism. The two chromosomes of the population are selected randomly and primarily the solution rank and secondarily the crowding distance are compared. A solution with a lower ranking and greater crowding distance is more favorable for selecting. After selecting the parents, the pairs are randomly selected and the offspring are generated by hyper-crossover. Hyper-crossover uses two types of one-point and two-point crossover operators randomly for each chromosome. This process will improve the evolution of the population. It should be noted there are four genotypes in a chromosome that each of them uses randomly one type of crossover to generate child. Fig. 5 shows the mode of one-point and two-point crossover operator.

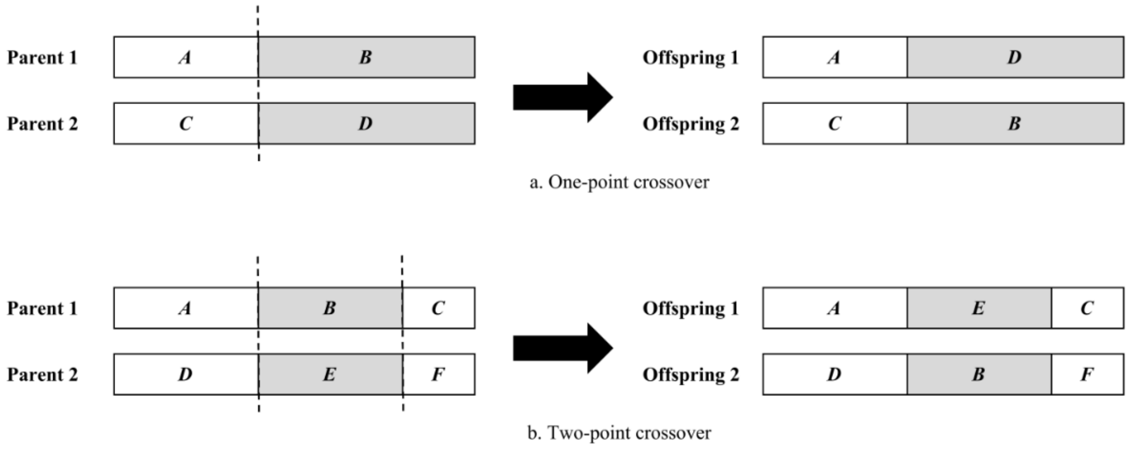

Figure 5. Hyper crossover scheme

Also, a small portion of the population, i.e. P_m, is mutated. Here an insert mutation is used for this operator. Fig. 6 shows the insertion mutation.

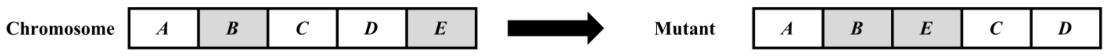

Figure 6. Insert mutation scheme

when using operators, the infeasible solutions may be generated that can be converted to a feasible solution by using the way presented in Section 3.3. After the generation of new offspring, these children are merged with the primary population, and with the non-dominated sorting engine, the solution ranks and crowding distances are obtained. With respect to these two criteria, the NP of the best solutions is selected as the next generation population. This process continues until it reaches the stop condition, i.e. the maximum iteration.

4. EXPERIMENTAL INVESTIGATIONS

In order to evaluate performance of the proposed methods, in this section a number of random generated instances is solved by these methods and various Pareto fronts are then created. To generate random instances, the benchmark problems presented by Kolisch and Sprecher (1997) are used. Activities, predecessors' relations, durations and the amount of renewable resources required are extracted from benchmark problems in the literature. Other parameters are generated randomly and through the presented formula in Table 5. It should be noted that the total amount of available renewable resources are determined by Resource Scarcity (RS) factor for each setting. This factor reflects the portion of average resources required per total available renewable resources. Clearly, with the increase of RS factor, the resource allocation planning becomes more difficult.

Table 5: Random generation of parameters

Parameters	Random generation formula
D_{ij}	Integer from uniform [0,5] for one project Integer from uniform [5,15] for different projects
s_{ij}	Integer from uniform [0,5] for one project Integer from uniform [5,10] for different projects
F_{lk}	Integer from uniform $[50, 200]$
r_{i2}	Integer from uniform $[0,10]$
α_i	Uniform $[0, 0.3]$
η_t	Periodic every 15 time units (i.e. $\eta_{15} = \eta_{30} = \eta_{45} = ... = 1$)
P_{kt}	Uniform $[800,1400] - \max\{0, (t - A_k)\} \times [1,5]$
CF_t	Uniform $[10, 20]$
CB_t	Uniform $[1, 3]$
CH	Uniform $[0.3, 1.2]$
CD	Uniform $[0.3, 1.2]$
CT_{ij}	Uniform $[1, 5]$
The number of potential location	3
The number of construction projects (*npr*)	3

In multi-objective algorithms, the metrics are used to evaluate quantitative performance. In this paper, three different metrics are defined which are as follows:

- Generational Distance (GD): This criterion shows the mean distance of the Pareto front obtained by the algorithm from the best obtained Pareto fronts. The zero value is favorable for GD metric. This metric is obtained by Eq. (31) in which dis_i is Euclidean distance of each solution *i* on the Pareto front obtained by the algorithm from the nearest solution in the best Pareto front, and *n* is the number of solutions in the Pareto front:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} dis_i^{\,2}}}{n} \tag{31}$$

- Maximum Spread (MS): This criterion shows the coverage of the best Pareto front by the Pareto front of the algorithm, which is calculated by Eq. (32):

$$MS = \sqrt{\tfrac{1}{2}\left[\left(\frac{\min\{f_{1,opt}^{\max}, f_1^{\max}\} - \max\{f_{1,opt}^{\min}, f_1^{\min}\}}{f_{1,opt}^{\max} - f_{1,opt}^{\min}}\right)^2 + \left(\frac{\min\{f_{2,opt}^{\max}, f_2^{\max}\} - \max\{f_{2,opt}^{\min}, f_2^{\min}\}}{f_{2,opt}^{\max} - f_{2,opt}^{\min}}\right)^2\right]} \tag{32}$$

In the above equation, $f_i^{\max}$ and $f_i^{\min}$ show the maximum and minimum value of the *i*-th objective function in the Pareto front of the algorithm. Also, $f_{i,opt}^{\max}$ and $f_{i,opt}^{\min}$ are the highest and lowest values of the *i*-th objective function in the best Pareto front. The favorable value for MS metric is 1.

- CPU Running Time (CRT): This criterion shows the time to reach the final Pareto front by each algorithm.

4.1 Parameters tuning mechanism

The performance of meta-heuristics methods depends heavily on their parameters. In the literature, there are various methods for tuning the parameters of these algorithms. In this paper, the Sum of Squares Error (SSE) index is used which was employed by Rostami et al. (2015).

In the NSABC algorithm the parameters NP, Iter_max, ω_1, ω_2 and Max_trial should be determined. Considering that in the PHA method, the number of members of the Pareto front is up to 64, so for the fair comparison between the algorithms, the NP value is considered to be 64. The value of the Iter_max is determined in the experiments of next section, and with this value, the parameter Max_trial is considered to be 5% of the Iter_max. For determining the appropriate values of the two important parameters ω_1 and ω_2 a full factorial design is conducted. The levels of these two parameters in the experiment are shown in Table 6.

Table 6: Factors and their levels for NSABC

Factors	Level
ω_1	0.6, 0.8, 1
ω_2	0.6, 0.8, 1

In order to tune these two parameters, 5 instances with varying dimensions and RS=0.2 are tested. Each instance will be run 3 times per each setting, and $\psi = GD/MS$ values for each run are reported. It should be noted that the best Pareto front for each instance is achieved with regard to the Pareto front obtained in all settings and selecting 64 members with the highest non-dominant rank. Based on these results, the best solution (BS), standard deviation (SD) and sum of square error (SSE) are calculated. The results of this experiment illustrate that the best setting for ω_1 and ω_2 are 0.6 and 1, respectively. Appendix 1 shows the details of the parameter setting of the NSABC algorithm.

In the NSGA-II the parameters NP, Iter_max, P_c and P_m should be determined. Considering that in the PHA method, the number of members of the Pareto front is up to 64, so for the fair comparison between the algorithms, the NP value is considered to be 64. The value of the Iter_max is determined in the experiments of next section. For determining the appropriate values of the two important parameters P_c and P_m a full factorial design is conducted. The levels of these two parameters in the experiment are shown in Table 7.

Table 7: Factors and their levels for NSGA-II

Factors	Level
P_c	40%, 50%, 60%
P_m	3%, 5%, 7%

In order to tune these two parameters, such as NSABC algorithm action, the best solution (BS), standard deviation (SD) and sum of square error (SSE) are calculated. The results of this experiment illustrate that the best setting for P_c and P_m are 50% and 7%, respectively. Appendix 2 shows the details of the parameter setting of the NSGA-II.

4.2 Comparison among existing methods

In this section, the proposed solution methods are evaluated. The mathematical programming model is coded by GAMS™ software. Also, heuristic and meta-heuristic algorithms are coded in the C # programming language. The proposed models are executed in a PC with CPU specifications of Intel Core i7 3.1 GHz and 8 GB of RAM and their results are reported.

Initially, the performance of heuristic methods is compared with ε-constraint method. Since the problem in this paper is a NP-hard problem, solving problems with the mathematical programming model is very time-consuming, and so this model is only used to solve small-size instances. In order to compare the performance of solving methods with the optimal Pareto front obtained by the mathematical model, 30 different instances are randomly generated and the results are presented in Table 8. These instances are categorized in 6 different setting. It should be noted that in order to reach the optimum Pareto front in a reasonable time (here is 18000 Sec.), the value of *m* is equal to 15, and, consequently, N_p is considered to be 15. Also, Fig. 7 shows the Pareto fronts obtained by solving methods for one of these instances with 15 activities and RS=0.15.

Table 8: The results for small-size instances

# of activities	RS factor	Method	Avg. of Indicators			# of unsolved instances by MIP
			GD	MS	CRT	
10	0.1	PHA	1.21	0.98	8.90	-
		NSABC	0.59	1.00	21.73	
		NSGA-II	0.85	1.00	16.42	
	0.15	PHA	1.38	0.97	9.17	-
		NSABC	0.87	1.00	23.52	
		NSGA-II	0.97	1.00	19.04	
	0.2	PHA	1.63	0.97	9.45	-
		NSABC	1.02	1.00	24.86	

		NSGA-II	1.09	1.00	20.61	
		PHA	1.42	0.97	15.56	
	0.1	NSABC	1.07	1.00	39.85	-
		NSGA-II	1.18	1.00	30.65	
		PHA	1.77	0.96	15.89	
15	0.15	NSABC	1.35	0.99	41.66	1
		NSGA-II	1.36	1.00	32.76	
		PHA	2.23	0.94	16.31	
	0.2	NSABC	1.54	0.98	43.17	3
		NSGA-II	1.71	0.98	34.16	

Comparison of the above results shows the appropriate performance of the proposed heuristic and meta-heuristic methods. As it is known, the Pareto front of the heuristic methods has little difference with the optimum Pareto front. Therefore, these heuristic methods can be relied on solving large-size instances.

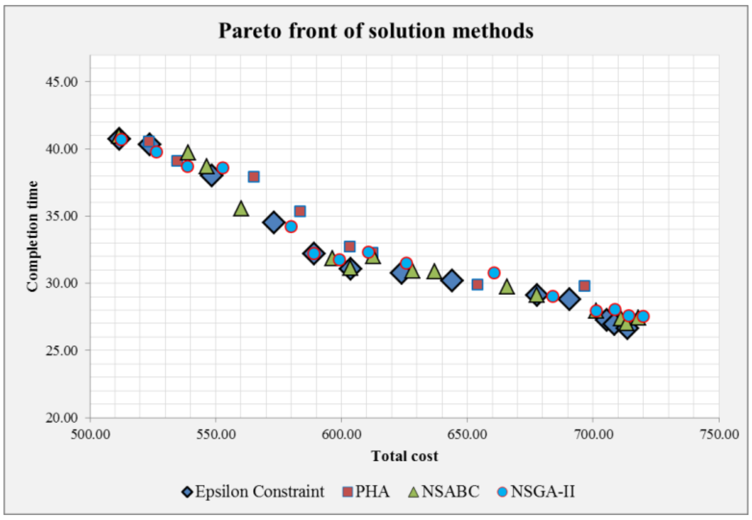

Figure 7. Pareto front of solution methods for an small-size instance with 15 activities and RS=0.15

In order to compare the performance of the proposed algorithms for large-size problems, 100 instances with different dimensions and setting are generated randomly (5 different random instances in each setting). These problems are solved using the algorithms proposed in this paper and the results of the defined metrics are presented in tables 9-11 accordingly. In these tables, the NSABC and NSGA-II algorithms are examined considering two modes associated with 100 and 200 iterations.

Table 9 shows the results of the GD index for the solution methods. As it is clear, by the average NSABC-200 algorithm has the lowest GD value. Also, the results show that by increasing the number of iterations, the NSABC method is improved over NSGA-II. The results show that the performance of the PHA algorithm is significantly reduced by increasing the dimensions of the problems under consideration.

Table 9: GD results for solving methods

# of activities	RS factor	PHA	NSABC-100	NSGA-II-100	NSABC-200	NSGA-II-200
30	0.1	4.68	3.96	3.86	3.53	3.58
	0.15	5.37	4.45	4.46	4.09	4.11
	0.2	5.99	4.49	4.85	4.12	4.33
	0.25	6.13	4.51	4.90	4.14	4.48
	0.3	6.85	5.29	5.47	4.39	4.69
60	0.1	23.98	16.51	17.32	15.92	16.88
	0.15	24.73	17.07	18.06	16.13	17.14
	0.2	25.31	17.11	18.38	16.21	17.09
	0.25	25.89	17.67	18.35	16.28	17.36
	0.3	27.16	18.16	18.52	16.49	17.93
90	0.1	45.44	28.37	30.34	27.04	29.53
	0.15	47.38	28.73	31.05	27.34	29.94
	0.2	48.16	29.14	31.29	28.74	30.20
	0.25	48.47	29.60	31.89	29.11	30.60
	0.3	49.61	29.68	31.84	29.52	30.65
120	0.1	102.12	56.55	60.83	48.66	56.92
	0.15	105.33	58.43	61.35	51.31	58.11
	0.2	109.87	59.03	61.94	52.26	58.49
	0.25	110.95	61.07	63.27	52.17	60.36
	0.3	115.95	62.05	65.14	54.19	61.50
Avg.		**46.97**	**27.59**	**29.16**	**25.08**	**27.69**

Table 10 shows the results of the MS index for the solution methods. As it is seen, by the average NSGA-II-200 algorithm has the highest MS value. Also, the results show that by increasing the number of iterations, the NSABC method is improved over NSGA-II. The results show that the performance of the PHA algorithm is significantly reduced by increasing the dimensions of the problems under consideration.

Table 10 MS results for solving methods

Instance No.	# of activities	PHA	NSABC-100	NSGA-II-100	NSABC-200	NSGA-II-200
30	0.1	0.97	0.98	1.00	1.00	1.00
	0.15	0.94	0.98	1.00	1.00	1.00
	0.2	0.91	0.98	1.00	1.00	1.00
	0.25	0.90	0.97	0.98	1.00	1.00
	0.3	0.88	0.97	0.95	0.99	0.99
60	0.1	0.90	1.00	1.00	1.00	1.00
	0.15	0.88	0.99	1.00	1.00	1.00

	0.2	0.87	0.98	1.00	1.00	1.00
	0.25	0.85	0.98	0.98	1.00	1.00
	0.3	0.83	0.96	0.98	0.99	0.98
	0.1	0.71	0.99	0.99	1.00	1.00
	0.15	0.68	0.98	0.99	0.99	1.00
90	0.2	0.67	0.96	0.98	0.98	0.99
	0.25	0.65	0.96	0.98	0.98	0.99
	0.3	0.65	0.93	0.97	0.97	0.98
	0.1	0.71	0.95	0.97	0.99	0.98
	0.15	0.69	0.94	0.96	0.98	0.98
120	0.2	0.66	0.94	0.97	0.97	0.97
	0.25	0.62	0.95	0.96	0.97	0.97
	0.3	0.61	0.93	0.95	0.97	0.96
Avg.		**0.780**	**0.966**	**0.981**	**0.989**	**0.990**

Finally, Table 11 shows the results of the CRT index for the solution methods. As can be seen, by the average PHA algorithm has the lowest CRT value. Also, the results show that by increasing the number of iterations, the CPU running time of NSABC method is increased faster than other existing methods. The results show that the performance of PHA algorithm remains acceptable for this metric by increasing the dimensions of the problems.

Table 11: CRT results for solving methods

Instance No.	# of activities	PHA	NSABC-100	NSGA-II-100	NSABC-200	NSGA-II-200
	0.1	41.78	117.87	109.25	223.16	207.96
	0.15	43.21	123.82	110.09	228.20	209.47
30	0.2	44.19	125.11	112.25	236.80	214.24
	0.25	46.19	128.32	113.47	239.44	218.00
	0.3	47.82	128.18	115.61	246.14	218.75
	0.1	143.79	302.82	278.39	621.99	568.05
	0.15	144.82	305.11	283.68	626.32	575.65
60	0.2	145.50	308.17	294.30	632.18	583.31
	0.25	147.68	307.03	293.59	635.08	598.02
	0.3	153.72	311.49	299.52	642.47	598.13
	0.1	224.66	526.36	467.27	1090.02	967.18
	0.15	228.66	531.57	480.15	1097.55	981.79
90	0.2	234.09	536.04	483.01	1107.30	998.03
	0.25	237.86	535.14	478.29	1102.89	991.72
	0.3	242.71	539.66	489.28	1113.30	1012.81
	0.1	403.28	973.41	859.55	1993.35	1735.34
	0.15	405.32	984.35	870.49	2004.90	1738.28
120	0.2	409.79	992.38	882.14	2029.31	1765.01
	0.25	415.09	1004.57	885.13	2045.85	1786.95
	0.3	433.55	1019.41	901.13	2074.24	1818.92
Avg.		**209.69**	**490.04**	**440.33**	**999.52**	**889.38**

4.3 Ranking the methods

In this section, five heuristic methods presented in this paper, namely PHA, NSABC-100, NSGA-II-100, NSABC-200 and NSGA-II-200 are ranked using a scenario-based TOPSIS method based on the defined metrics. Considering that in multi-criteria ranking methods, the significance of each criterion depends on the expert's judgment, thus ranking of solution methods is performed in 18 different scenarios based on different weights of the metrics. Table 12 shows the weights of the metrics at each scenario. These weights are adjusted so that the variability of the significance of each metric is met in comparison with other metrics.

Table 12: Weight of metrics based on scenarios

Metrics	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
GD	0.8	0.8	0.8	0.6	0.6	0.4	0.1	0.2	0.0	0.1	0.3	0.3	0.1	0.0	0.2	0.3	0.1	0.3
MS	0.1	0.2	0.0	0.3	0.1	0.3	0.8	0.8	0.8	0.6	0.6	0.4	0.1	0.2	0.0	0.1	0.3	0.3
CRT	0.1	0.0	0.2	0.1	0.3	0.3	0.1	0.0	0.2	0.3	0.1	0.3	0.8	0.8	0.8	0.6	0.6	0.4

Then, the algorithms are ranked for each scenario using the TOPSIS method. The TOPSIS approach is implemented for all scenarios and SW_i values for each algorithm in each scenario are reported in Table 13.

Table 13: SW values for all scenarios

Methods	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
PHA	0.18	0.00	0.30	0.22	0.46	0.56	0.38	0.00	0.57	0.70	0.33	0.61	0.93	0.95	0.87	0.77	0.88	0.69
NSABC-100	0.87	0.89	0.83	0.86	0.76	0.72	0.80	0.89	0.72	0.68	0.82	0.70	0.65	0.65	0.65	0.66	0.65	0.68
NSGA-II-100	0.81	0.81	0.79	0.81	0.76	0.75	0.84	0.88	0.78	0.73	0.81	0.74	0.71	0.71	0.71	0.72	0.71	0.73
NSABC-200	0.82	1.00	0.70	0.78	0.54	0.44	0.62	1.00	0.43	0.30	0.67	0.39	0.07	0.05	0.13	0.23	0.12	0.31
NSGA-II-200	0.80	0.88	0.69	0.77	0.54	0.45	0.65	0.93	0.47	0.34	0.68	0.40	0.15	0.15	0.18	0.25	0.18	0.33

As a result, Table 14 shows the rank of the algorithms for each scenario.

Table 14 ranking of methods for all scenarios

Methods	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
PHA	5	5	5	5	5	3	5	5	3	2	5	3	1	1	1	1	1	2
NSABC-100	1	2	1	1	2	2	2	3	2	3	1	2	3	3	3	3	3	3
NSGA-II-100	3	4	2	2	1	1	1	4	1	1	2	1	2	2	2	2	2	1
NSABC-200	2	1	3	3	4	5	4	1	5	5	4	5	5	5	5	5	5	5
NSGA-II-200	4	3	4	4	3	4	3	2	4	4	3	4	4	4	4	4	4	4

The results of Table 14 show that the NSABC algorithm is more suitable for situations in which the GD metric is more important (scenarios 1 to 6). Also, when the MS metric is the most important index (scenarios 7 to 12), the NSGA-II algorithm is recommended. Finally, when the CRT metric is considered by decision maker (scenarios 13 to 18) the PHA will have the best performance.

Based on the results of Table 14, the Friedman test can be used to rank algorithms. The Friedman test is one of the nonparametric tests, which without the need for a normality assumption, as an alternative to the ANOVA method, identifies the same or different rank of the methods. In the case of different ranking algorithms (reject the null hypothesis), paired comparison of methods is necessary. For this purpose Wilcoxon signed ranks test can be used. Table 15 shows the results of Friedman test.

Table 15: Friedman test result

Methods	Mean Rank	Chi-Square	df	Asymp. Sig.
PHA	3.22	24.000	4	0.000
NSABC-100	2.22			
NSGA-II-100	1.89			
NSABC-200	4.00			
NSGA-II-200	3.67			

According to the Table 15, it is clear that the null hypothesis of Friedman test is rejected. For this reason, the paired comparisons are conducted by Wilcoxon signed ranks test and its results are presented in Table 16.

Table 16: Wilcoxon signed ranks test result

	PHA - NSABC100	PHA - NSGAII100	PHA - NSABC200	PHA - NSGAII200	NSABC100 - NSGAII100	NSABC100 - NSABC200	NSABC100 - NSGAII200	NSGAII100 - NSABC200	NSGAII100 - NSGAII200	NSABC200 - NSGAII200
Z	-1.625[a]	-2.678[a]	-1.210[b]	-.948[b]	-1.039[a]	-3.410[b]	-3.562[b]	-2.903[b]	-3.342[b]	-1.189[a]
Asymp. Sig. (2-tailed)	.104	.007	.226	.343	.299	.001	.000	.004	.001	.234

a. Based on negative ranks.
b. Based on positive ranks.

Thus Table 17 shows the result of the paired comparison of solution methods.

Table 17: Paired comparison of the algorithms

Rank	Methods	NSGA-II-100	NSABC-100	PHA	NSGA-II-200	NSABC-200
1	NSGA-II-100	-	No difference	Significant difference	Significant difference	Significant difference
2	NSABC-100		-	No difference	Significant difference	Significant difference
3	PHA			-	No difference	No difference
4	NSGA-II-200				-	No difference
5	NSABC-200					-

The computational experiments presented in this paper can be concluded as follows:

1- From the GD index’s point of view, the NSABC-200 algorithm has shown the best performance.

2- According to GD index, the performance of the PHA algorithm is significantly reduced by increasing the dimensions of the problems.
3- From the point of view of MS index, the NSGA-II-200 algorithm has shown the best performance.
4- According to MS index, the performance of the PHA algorithm is significantly reduced by increasing the dimensions of the problems.
5- From the point of view of CRT index, the PHA algorithm has shown the best performance.
6- By increasing the number of iterations, the CPU running time of NSABC method is increased faster than other existing heuristic methods.
7- According to CRT index, the performance of PHA algorithm remains acceptable by increasing the dimensions of the problems.
8- For the ranking of the algorithms, a scenario-based TOPSIS method is used. The results of Friedman's test and Wilcoxon signed ranks show that the NSGA-II-100 has totally the highest rank according to defined scenarios, followed by NSABC-100 and PHA in the following ranks.
9- From decision maker's point of view, the NSABC algorithm is more suitable for situations in which the GD metric is more important. Also, when the MS metric is the most important index, the NSGA-II algorithm is recommended. Finally, when the CRT metric is more important, the PHA will have the best performance.

5. CONCLUSION REMARK AND FURTHER RECOMMENDATIONS

In this paper, a multi-objective decentralized multiple construction projects scheduling problem with considering periodic services of renewable resources and ordering policies of non-renewable resources was proposed. The first objective function was to minimize the completion time of the construction projects, and the second objective function was to minimize the total cost of project implementation. In order to describe the constraints and solving small-size problems, a mixed integer linear programming model was proposed using ε-constraint method, which can obtain the optimal Pareto front. Also, by incorporating a new version of the Parallel Schedule Generation Scheme (PSGS), a Priority-rule based Heuristic Algorithm (PHA); a Non-dominate Sorting Artificial Bee Colony (NSABC) algorithm and a Non-dominate Sorting Genetic Algorithm (NSGA-II) were also developed.

By defining three metric GD, MS and CRT, the solution methods were evaluated. With the aim of obtaining the lowest GD, the results showed that the best known solution method is NSABC. The most appropriate method for optimizing the MS metric is the NSGA-II. Also the PHA is the fastest solution method (the best CRT). The computational results showed that increasing the number of iterations has no significant effect on the improvement of the Pareto front, and therefore it is recommended to set the iteration number over 100.

Finally, using TOPSIS approach, the proposed heuristic methods were ranked. Based on the 18 scenarios defined in terms of weight of metrics, the decision maker can choose the best known solution method. The results of Friedman and Wilcoxon signed ranks tests showed that NSGA-II-100 has totally the highest rank according to defined scenarios, followed by NSABC-100 and PHA in the following ranks. Since this paper has dealt with deterministic construction project information, future research can be conducted on fuzzy multi objective project scheduling problem under periodic services.

Conflict of interest statement
On behalf of all authors, the corresponding author states that there is no conflict of interest.

APPENDIX 1: PARAMETERS TUNING OF NSABC IS SHOWN IN TABLE 17

Table 17: Parameters setting of NSABC

Instances	ω_1	ω_2	**Run**			**BS**	**SD**	**SSE**
			Run 1	Run 2	Run 3			
15j-3p	0.6	0.6	1.49	1.92	1.59	1.15	1.0	0.90
		0.8	1.44	1.27	1.27		0.3	0.11
		1	1.19	1.15	1.28		0.1	0.02
	0.8	0.6	2.47	1.57	1.27		1.4	1.92
		0.8	2.24	2.52	2.00		1.9	3.77
		1	1.79	1.92	2.61		1.8	3.12
	1	0.6	2.60	2.11	2.68		2.3	5.37
		0.8	2.10	2.33	1.94		1.7	2.93
		1	2.00	1.82	2.39		1.6	2.71
30j-3p	0.6	0.6	5.08	4.79	5.62	4.69	1.0	1.02
		0.8	5.50	5.07	4.92		0.9	0.85
		1	4.97	4.69	5.21		0.6	0.34
	0.8	0.6	5.11	5.14	5.01		0.7	0.48
		0.8	5.40	5.60	5.36		1.3	1.78
		1	5.48	5.20	5.74		1.4	1.98
	1	0.6	5.23	5.21	5.40		1.0	1.07
		0.8	5.21	4.95	5.19		0.8	0.59
		1	5.26	5.15	4.93		0.8	0.60
60j-3p	0.6	0.6	18.96	20.24	19.37	17.65	3.4	11.34
		0.8	17.99	19.25	20.56		3.3	11.14
		1	19.53	17.65	18.05		1.9	3.68
	0.8	0.6	18.27	19.97	18.60		2.6	6.64
		0.8	20.53	18.22	19.77		3.6	13.08
		1	18.76	20.11	18.02		2.7	7.42
	1	0.6	22.14	19.52	20.67		5.7	32.73
		0.8	19.62	19.13	19.76		3.2	10.52
		1	19.91	19.71	18.91		3.3	10.94
90j-3p	0.6	0.6	31.65	35.37	34.24	30.73	5.9	34.78
		0.8	32.88	33.17	31.54		3.4	11.29
		1	32.16	33.52	30.73		3.1	9.87
	0.8	0.6	34.13	33.74	34.73		6.1	36.72
		0.8	31.70	34.88	32.74		4.7	22.26

		1	34.11	32.69	32.35		4.2	17.94
		0.6	35.20	33.19	31.57		5.2	26.82
	1	0.8	37.56	36.18	34.26		9.4	88.88
		1	34.97	31.08	33.68		5.2	26.85
		0.6	77.55	70.62	83.90		21.7	471.97
	0.6	0.8	75.94	66.05	82.26		19.0	360.77
		1	73.07	73.95	69.55		11.1	124.22
		0.6	83.19	76.83	69.29		20.5	420.82
120j-3p	0.8	0.8	79.15	70.82	80.12	66.05	19.8	392.88
		1	71.27	75.07	70.16		11.2	125.63
		0.6	76.13	66.37	73.81		12.7	162.15
	1	0.8	88.58	75.31	71.58		25.0	624.19
		1	68.10	80.82	72.29		16.2	261.31

APPENDIX 2: PARAMETERS TUNING OF NSGA-II IS SHOWN IN TABLE 18

Table 18: Parameters setting of NSGA-II

Instances	P_c	P_m	**Run**			**BS**	**SD**	**SSE**
			Run 1	Run 2	Run 3			
		3%	2.24	2.11	2.10		1.8	3.11
	40%	5%	2.23	2.38	2.03		1.9	3.59
		7%	1.92	1.92	1.94		1.4	1.89
		3%	1.56	1.57	2.21		1.2	1.54
15j-3p	50%	5%	1.50	1.51	1.17	1.13	0.5	0.28
		7%	1.28	1.13	1.35		0.3	0.07
		3%	1.52	1.46	1.31		0.5	0.29
	60%	5%	1.30	1.45	1.72		0.7	0.47
		7%	1.17	1.23	1.41		0.3	0.09
		3%	5.71	6.15	6.11		2.5	6.38
	40%	5%	5.63	6.30	5.69		2.4	5.55
		7%	5.23	5.62	5.97		1.9	3.65
		3%	5.29	4.89	4.95		0.9	0.83
30j-3p	50%	5%	4.87	4.55	4.84	4.55	0.4	0.19
		7%	4.66	4.70	4.62		0.2	0.04
		3%	5.07	4.97	4.79		0.7	0.52
	60%	5%	4.93	4.66	4.66		0.4	0.17
		7%	4.59	4.78	4.72		0.3	0.09
		3%	23.21	23.57	23.99		9.6	91.94
	40%	5%	22.17	23.28	23.05		8.3	68.97
		7%	22.67	20.43	21.55		6.2	39.01
60j-3p		3%	20.59	21.02	19.76	18.06	4.2	18.03
	50%	5%	19.15	19.26	18.75		1.8	3.10
		7%	18.63	18.06	18.77		0.9	0.83
	60%	3%	19.31	20.05	19.94		3.0	9.05

		5%	19.18	19.01	19.30		1.9	3.70
		7%	18.56	19.04	18.43		1.2	1.35
		3%	41.89	40.16	41.09		16.0	254.50
	40%	5%	38.22	38.82	39.60		12.2	148.51
		7%	38.66	38.24	36.67		10.5	109.94
		3%	35.33	35.91	33.70		5.6	31.77
90j-3p	50%	5%	34.04	33.53	34.18	31.86	3.6	12.85
		7%	32.35	33.64	33.17		2.3	5.09
		3%	33.77	33.48	33.78		3.1	9.91
	60%	5%	32.07	34.04	33.08		2.5	6.25
		7%	33.18	31.86	33.86		2.4	5.72
		3%	96.37	90.40	93.00		42.7	1825.90
	40%	5%	89.93	87.80	83.60		32.2	1036.81
		7%	81.67	89.87	82.89		28.6	817.30
		3%	74.02	76.92	77.20		12.9	167.70
120j-3p	50%	5%	76.59	72.81	75.42	68.71	11.1	123.95
		7%	71.16	74.38	74.62		8.6	73.11
		3%	75.82	73.71	78.51		13.1	171.57
	60%	5%	78.75	68.71	74.76		11.7	137.42
		7%	77.59	68.89	69.60		8.9	79.65

REFERENCES

1. Adhau S, Mittal M, Mittal A. A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach, *Eng Applicat Artificial Intell* 2012; **25**: 1738-51.
2. Akay B. Synchronous and asynchronous Pareto-based multi-objective Artificial Bee Colony algorithms, *J Global Optim* 2013; **57**: 415-45.
3. Akbari R, Hedayatzadeh R, Ziarati K, Hassanizadeh B. A multi-objective artificial bee colony algorithm, *Swarm Evolut Comput* 2012; **2**: 39-52.
4. Almeida BF, Correia I, Saldanha-Da-Gama F. A biased random-key genetic algorithm for the project scheduling problem with flexible resources, *Top* 2018: 1-26.
5. Alnasseri H, Aulin R.. Assessing understanding of planning and scheduling theory and practice on construction projects, *Eng Manag J* 2015; **27**: 58-72.
6. Aquilano NJ, Smith DE. A formal set of algorithms for project scheduling with critical path scheduling/material requirements planning, *J Operat Manag* 1980; **1**: 57-67.
7. Beşikci U, Bilge Ü, Ulusoy G. Resource dedication problem in a multi-project environment, *Flexible Serv Manufact J* 2013; **25**: 206-29.
8. Beşikci U, Bilge Ü, Ulusoy G. Multi-mode resource constrained multi-project scheduling and resource portfolio problem, *European J Operat Res* 2015; **240**: 22-31.
9. Bilolikar VS, Jain K, Sharma M. An adaptive crossover genetic algorithm with simulated annealing for multi mode resource constrained project scheduling with discounted cash flows, *Int J Operat Res* 2016; **25**: 28-46.
10. Can A, Ulusoy G. Multi-project scheduling with two-stage decomposition, *Annals*

Operat Res 2014; **217**: 95-116.

11. Capa C, Ulusoy G. Proactive project scheduling in an R&D department a bi-objective genetic algorithm, *Industrial Engineering and Operations Management (IEOM), International Conference on* 2015; *IEEE,* pp. 1-6.
12. Çebi F, Otay İ. A fuzzy multi-objective model for solving project network problem with bonus and incremental penalty cost, *Comput Indust Eng* 2015; **82**: 143-50.
13. Chakrabortty RK, Sarker RA, Essam DL. Resource constrained multi-project scheduling: a priority rule based evolutionary local search approach, *Intell Evolut Syst* 2017; 75.
14. Chen R, Liang C, Gu D, Leung JY. A multi-objective model for multi-project scheduling and multi-skilled staff assignment for IT product development considering competency evolution, *Int J Product Res* 2017; **55**: 6207-34.
15. Chen W, Lei L, Wang Z, Teng M, Liu J. Coordinating supplier selection and project scheduling in resource-constrained construction supply chains, *Int J Product Res* 2018; **56**: 6512-26.
16. Davari M, Demeulemeester E. A novel branch-and-bound algorithm for the chance-constrained resource-constrained project scheduling problem, *Int J Product Res* 2019; **57**: 1265-82.
17. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transact Evolut Comput* 2002; **6**: 182-97.
18. Dodin B, Elimam A. Integrated project scheduling and material planning with variable activity duration and rewards, *IIE Transactions* 2001; **33**: 1005-18.
19. El-Abbasy MS, Elazouni A, Zayed T. Moscopea: Multi-objective construction scheduling optimization using elitist non-dominated sorting genetic algorithm, *Automat Construct* 2017; **71**: 153-70.
20. Fu F. Integrated scheduling and batch ordering for construction project, *Appl Mathemat Modell* 2014; **38**: 784-97.
21. Geiger MJ. A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem, *European J Operat Res* 2017; **256**: 729-41.
22. Ghoddousi P, Eshtehardian E, Jooybanpour S, Javanmardi A. Multi-mode resource-constrained discrete time–cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm, *Automat Construct* 2013; **30**, 216-27.
23. Gomes HC, Das Neves FDA, Souza MJF. Multi-objective metaheuristic algorithms for the resource-constrained project scheduling problem with precedence relations, *Comput Operat Res* 2014; **44**: 92-104.
24. Gutjahr WJ. Bi-objective multi-mode project scheduling under risk aversion, *European J Operat Res* 2015; **246**: 421-34.
25. He J, Chen X, Chen X. A Filter-and-fan approach with adaptive neighborhood switching for resource-constrained project scheduling, *Comput Operat Res* 2016; **71**: 71-81.
26. Holland JH. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, U Michigan Press, 1975.

27. Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J Global Optim* 2007; **39**: 459-71.
28. Kaveh A, Khanzadi M, Alipour M. Fuzzy resource constraint project scheduling problem using CBO and CSS algorithms, *Int J Civil Eng* 2016; **14**: 325-37.
29. Kishor A, Singh PK, Prakash J. NSABC: Non-dominated sorting based multi-objective artificial bee colony algorithm and its application in data clustering, *Neurocomput* 2016; **216**: 514-33.
30. Kolisch R, Sprecher A. PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program, *European J Operat Res* 1997; **96**: 205-16.
31. Krüger D, Scholl A. A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer times, *European J Operat Res* 2009; **197**: 492-508.
32. Krüger D, Scholl A. Managing and modelling general resource transfers in (multi-) project scheduling, *OR Spectrum* 2010; **32**: 369-94.
33. Küçüksayacigil F, Ulusoy G. A genetic algorithm application for multi-objective multi-project resource constrained project scheduling problem, 2014.
34. Lacomme P, Moukrim A, Quilliot A, Vinot M. A new shortest path algorithm to solve the resource-constrained project scheduling problem with routing from a flow solution, *Eng Applicat Artific Intell* 2017; **66:** 75-86.
35. Laszczyk M, Myszkowski PB. Improved selection in evolutionary multi–objective optimization of Multi–Skill Resource–Constrained project scheduling problem, *Inform Sci* 2019; **481,** 412-31.
36. Lu H, Zhou R, Fei Z, Shi J. A multi-objective evolutionary algorithm based on Pareto prediction for automatic test task scheduling problems, *Appl Soft Comput* 2018; **66:** 394-412.
37. Myszkowski PB, Olech ŁP, Laszczyk M, Skowroński ME. Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem, *Appl Soft Comput* 2018; **62:** 1-14.
38. Myszkowski PB, Skowroński ME, Olech ŁP, Oślizło K. Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem, *Soft Comput* 2015; **19**: 3599-619.
39. Rostami M, Bagherpour M. A lagrangian relaxation algorithm for facility location of resource-constrained decentralized multi-project scheduling problems, *Operat Res* 2020; **20**: 857-97.
40. Rostami M, Bagherpour M, Mazdeh MM, Makui A. Resource pool location for periodic services in decentralized multi-project scheduling problems, *J Comput Civil Eng* 2017; **31**: 04017022.
41. Rostami M, Kheirandish O, Ansari N. Minimizing maximum tardiness and delivery costs with batch delivery and job release times, *Appl Mathemat Modell* 2015; **39**: 4909-27.
42. Rostami M, Moradinezhad D, Soufipour A. Improved and competitive algorithms for large scale multiple resource-constrained project-scheduling problems, *KSCE J Civil Eng* 2014; **18:** 1261-9.

43. Sajadieh MS, Shadrokh S, Hassanzadeh F. Concurrent project scheduling and material planning: A genetic algorithm approach, *Scientia Iranica* 2009; **16**: 91-9.
44. Salewski F, Schirmer A, Drexl A. Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application, *European J Operat Res* 1997; **102**: 88-110.
45. Sonmez R, Uysal F. Backward-forward hybrid genetic algorithm for resource-constrained multiproject scheduling problem, *J Comput Civil Eng* 2014; **29**: 04014072.
46. Tavana M, Abtahi AR, Khalili-Damghani K. A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems, *Expert Syst Applicat* 2014; **41**: 1830-46.
47. Tirkolaee EB, Goli A, Hematian M, Sangaiah AK, Han T. Multi-objective multi-mode resource constrained project scheduling problem using Pareto-based algorithms, *Comput* 2019; **101**: 547-70.
48. Tran DH, Cheng MY, Cao MT. Solving resource-constrained project scheduling problems using hybrid artificial bee colony with differential evolution, *J Comput Civil Eng* 2015; 04015065.
49. Van Peteghem V, Vanhoucke M. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances, *European J Operat* Res 2014; **235:** 62-72.
50. Wang J, Hu X, Demeulemeester E, Zhao Y. A bi-objective robust resource allocation model for the RCPSP considering resource transfer costs, *Int J Product Res* 2019; 1-21.
51. Wang L, Zheng XL. A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem, *Swarm Evolut Comput* 2018; **38:** 54-63.
52. Wang WX, Wang X, Ge XL, Deng L. Multi-objective optimization model for multi-project scheduling on critical chain, *Adv Eng Soft* 2014; **68:** 33-9.
53. Yang KK, Sum CC. A comparison of resource allocation and activity scheduling rules in a dynamic multi-project environment, *J Operat Manag* 1993; **11**: 207-18.
54. Zhong YB, Xiang Y, Liu HL. A multi-objective artificial bee colony algorithm based on division of the searching space, *Appl Intell* 2014; **41:** 987-1011.
55. Zoraghi N, Shahsavar A, Abbasi B, Van Peteghem V. Multi-mode resource-constrained project scheduling problem with material ordering under bonus–penalty policies, *Top* 2017; **25**: 49-79.