Journal of Industrial and Systems Engineering
Vol. 11, No. 3, pp. 84-101
Summer (July) 2018

JISE

Project resource investment problem under progress payment model

Elham Merrikhi[1], Amir Abbas Najafi[2*], Aria Shahsavar[3]

[1]Research School of Management, Australian National University, Canberra, Australia
[2]Faculty of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran
[3]Faculty of Industrial and Mechanical Engineering, Islamic Azad University, Qazvin Branch, Qazvin, Iran

elhammerrikhi@yahoo.com, aanajafi@kntu.ac.ir, m.shahsavar@ymail.com

Abstract

As a general branch of project scheduling problems, resource investment problem (RIP) considers resource availabilities as decision variables to determine a level of employed resources minimizing the costs of the project. In addition to costs (cash outflows), researchers in the later extensions of the RIP took payments (cash inflows) received from clients into account and used the net present value (NPV) of project cash flows as a financial criterion evaluating the profitability of the project. A striking point in a financial view of the project scheduling is how cash inflows are paid by client. There are different payment models in the literature of which progress payment is highly common in practice. In this paper, resource investment problem with maximization of the NPV under progress payment model is investigated. A new mathematical model is developed for the problem and then two metaheuristic algorithms based on the genetic algorithm (GA) and simulated annealing algorithm (SA) is suggested. The experimental results of algorithms are compared with some near optimal solutions derived from LINGO software. The comparisons show that the results of GA are more reliable than SA.

Keywords: Project scheduling, Resource investment, Net present value, Progress payment.

1-Introduction and Literature review

Introduced in the late 1950s, the project scheduling problem has been widely considered by researchers in different views resulting in different branches. One of the well-researched branches is resource leveling problem (RLP) which aims at minimization of the variation of the resource requirements during the project life cycle (Ahuja, 1976). The resource constrained project scheduling problem (Blazewicz et al., 1983) is the most well-known and researched model in project scheduling which minimizes the make-span of a project at scarce resource situation. The Resource investment problem is another well-known branch introduced by Mohring (1984) in which the resource availability is considered as decision variable with aim of finding a schedule for activities together with the availability of resources minimizing the renewable resource costs of the project.

*Corresponding author
ISSN: 1735-8272,

Demeulemeester (1995) presented an exact algorithm for RIP based on a branch and bound method previously designed for resource constrained project scheduling problem. Zimmermann and Engelhardt (1998) developed another branch and bound procedure for RIP under generalized precedence relations (RIP/max). Nübel (1999) suggested another method for RIP/max with the help of fictitious resource capacities and the resolution of resulting resource conflicts. Later on, Nübel (2001) developed a depth-first branch-and-bound method for solving a generalized RIP/max. In addition, Drexel and Kimms (2001) employed Lagrangian relaxation for finding the lower and upper bound for the RIP. As for metaheuristic procedures, Yamashita et al. (2006) proposed a scatter search, Shadrokh and Kianfar (2007) suggested a GA where the penalty would be forced to the project if the tardiness is met, and Ranjbar et al. (2008) introduced a path relinking and genetic algorithm procedure to tackle the RIP. Sabzehparvar et al. (2008) proposed a mathematical model for the RIP with multi-execution activities. Rodrigues and Yamashita (2010) proposed a hybrid method in which a heuristic algorithm feeds a branching approach to find optimal solution. Afshar-Nadjafi (2014) developed a mixed integer programming formulation for the multi-mode RIP wherein resources have recruitment and release dates. They also proposed a simulated annealing problem. Qi et al. (2015) designed a hybrid particle swarm optimization and scatter search to find satisfying solutions for the multi-mode RIP. Shahsavar et al. (2016) studied an integration of RIP and material ordering and proposed three hybrid metaheuristics. Javanmard et al. (2016) added multi-skilled view to the RIP to minimize total cost of recruiting different levels of skills. Two metaheuristic algorithms with calibrated parameters were also designed.

The integration of project scheduling and financial indexes was firstly done by Russel (1970) who considered the maximization of the project's cash flows as the objective. Grinold (1972) presented an algorithm to evaluate the exact trade-off between net present value and project completion time. Smith-Daniels and Aquilano (1987) considered Russell's work in the presence of the time for occurring positive and negative cash flows of the project. They illustrated an example in which the NPV is maximized by scheduling all non-critical activities with negative cash flow at the latest possible time and scheduling activities with positive cash flows at the earliest possible time. Elmaghraby and Herroelen (1990) developed an approach for solving max-NPV problems without resource constraints in an AOA network and fixed cash flow. Later, Herroelen and Gallens (1993) showed more computational experience for the work of Elmaghraby and Herroelen (1990). Etgar et al. (1996) formulated a problem of time-dependent cash flows to maximize the NPV. In their model, the bonus and penalty for the early and late events were considered. In addition, the costs were assumed to be changing over time. Shtub and Etgar (1997) proposed a branch and bound algorithm for the problem of NPV maximization of time-dependent cash flows in project scheduling. Doersch and Patterson (1997) presented a zero-one mathematical model in the condition of limited budget. Vanhoucke et al. (1999) considered the non-increasing linear cash flows in an AON network without resources constraints and in a condition where negative cash flows are occurred at the end of activities. They suggested a backtracking search algorithm in which the entire project activities are scheduled in the earliest possible time and then identified the series of activities that can maximize NPV by a backtracking search and then pushes the activities forwards as much as permitted. Etgar and Shtub (1999) assumed that the cash flows are linear functions of the events realization time and developed an optimal solution for the problem. Najafi and Niaki (2006) introduced the integration of RIP and max-NPV and proposed a GA for the problem. Later on, Najafi et al. (2009) studied RIP/max under max-NPV and presented a parameter-tuned GA. Najafi and Azimi (2009) permitted the problem to be tardy with penalty and proposed heuristic approaches. Shahsavar et al. (2011) assessed RIP/max with discounted cash flows under inflationary conditions where earliness and tardiness is also permitted with a bonus and penalty.

An essential part of real projects that is stipulated in the contracts is how the contractor receives the payments or cash outflows from the client. In the literature, there are four types of payment models. In Lump-sum payment (LSP) type, the whole payment is received at the completion time of the project. In the payments at event occurrence (PEO) type, some events of the project are predetermined for receiving the payment. In the equal time interval (ETI) type, some equal intervals throughout the project life cycle are determined for occurrence of payments. In the progress payment (PP) type, the client pays the

payments at equal intervals, for example monthly, of the project duration based on the progress of the project i.e. the work accomplished during that interval. Despite of being the most common type of payment in the real world situations, the progress payment has not been seriously taken into account by researchers so far. For example, none of research above have considered the progress payment. Among a limited number of efforts done we can refer to Kazaz and Sepil (1996) who presented a mixed integer programing for max-NPV problem with progress payment. Sepil and Ortac (1997) extended progress payment in resource constrained project scheduling and proposed three different heuristics to solve the model. Totally, where financial characteristics of a project are under deep consideration, the assessment of progress payment would be a necessary and open area for discussion and evaluation. Particularly, there is a real research gap in the RIP to be assessed under progress payment type leading to a more practical model. Therefore, what this paper is focusing on is to combine resource investment problem with maximization of NPV and the progress payment. In section 2, the mathematical formulation of the problem will be discussed. Owing to the NP-hardness of the problem, two meta-heuristic algorithms based on genetic algorithm and simulated annealing are taken into picture in section 3. Section 4 concerns with the computational results of algorithms and finally the conclusion of the study comes in section 5.

2-Problem formulation

A project is given with a set of n activities indexed from 1 to n where activities 1 and n are dummies representing the start and completion event of the project, respectively. In order to execute activities K types of renewable resources are required. It is assumed that, the resources are not available at the starting point of the project and each resource is recruited when needed for the first time and is expulsed when there is no activity left requiring that resource. Between the recruiting and the expulsion time of each resource, its availability level is fixed to the maximum level of demands during the corresponding period.
We impose zero-lag finish-to-start precedence constraints on the sequence of the activities. For each activity i, its predecessor and successor activity sets are denoted as $P(i)$ and $Suc(i)$, respectively. d_i denotes the duration of activity i where no preemption is permitted. Activity i uses r_{ik} units of resource k per period. The resource usage over an activity is taken to be uniform. The project is charged with a cost of C_k for one unit of recruited resource k per period of time. It is assumed that each activity i has a cash outflow c_i^- and a profit margin γ_i. The revenue from this activity i then amounts to $c_i^+ = c_i^-(1+\gamma_i)$. We further assume that the cash outflows c_i^- occur at the completion of the activities. The cash inflows (receipt from employer), c_i^+, are incurred as progress payments at the end of some time periods. These progress payments are received at fixed time points for the work completed (w_{it}) during the current period, i.e. for the finished and partially finished activities where w_{it} (i =1,2,.., n and $t = T, 2T, ..., mT$) denotes the portion of work completed during period [t-T, t) for activity i. The activities are to be scheduled such that the makespan of the project does not exceed a given due date (DD). Furthermore, we assume that the discount rate is α.
To formulate the problem, let us define the decision variables and symbols used as follows:
S_i: starting time of activity i, i = 1,2,..., n.
R_k: maximum required level of resource k to be recruited (to be constant), k = 1,2,..., K.
SR_k: recruiting time of resource k, k = 1,2,..., K.
FR_k: expulsion time of resource k, k = 1,2,..., K.
X_{it}: A binary variable where it is one if activity i starts at period t and zero otherwise, i = 1,2,..., n, and t = 0,1, …, DD.
$PR(k)$: the set of activities that uses resource k but none of their predecessors use this resource, k = 1,2,..., K.
$UR(k)$: the set of activities that uses resource k but none of their successors use this resource, k = 1,2,..., K.
ES_i: earliest start time of activity i. i = 1,2,..., n.
LS_i: latest start time of activity i. i = 1,2,..., n.
y_{it}: A binary variable where it is one if activity i is in progress at period t and zero otherwise, i = 1,2,..., n, and t = 0,1, …, DD.

We can now formulate the model as follows:

$$Max \quad Z = \sum_{i=1}^{n} \sum_{t=T,2T,...,mT} w_{it} c_i^{+} e^{-\alpha t} - \sum_{i=1}^{n} c_i^{-} e^{-\alpha (S_i + d_i)} - \sum_{k=1}^{K} C_k R_k \left(\frac{e^{-\alpha SR_k} - e^{-\alpha FR_k}}{1 - e^{-\alpha}} \right) \tag{1}$$

$$S_i - S_j \geq d_j \quad ; \forall j \in P(i), \; i = 1,2,...,n \tag{2}$$

$$S_n \leq DD \tag{3}$$

$$SR_k \leq S_i \quad ; \quad \forall i \in PR(k), \; k = 1,2,...,K \tag{4}$$

$$FR_k \geq S_i + d_i \quad ; \quad \forall i \in UR(k), \; k = 1,2,...,K \tag{5}$$

$$\sum_{i=1}^{n} \sum_{l=t-d_i+1}^{t} r_{ik} X_{il} \leq R_k \quad ; \; t = 0,1,2,...,DD-1, \; k = 1,2,...,K \tag{6}$$

$$\sum_{t=ES_i}^{LS_i} X_{it} = 1 \quad ; i = 1,2,...,n \tag{7}$$

$$S_i = \sum_{t=ES_i}^{LS_i} t X_{it} \quad ; i = 1,2,...,n \tag{8}$$

$$\sum_{t=\max(0,l-d_i+1)}^{l} X_{it} \leq y_{il} \quad ; i = 1,2,...,n; \quad l = 0,1,...,DD-1 \tag{9}$$

$$\sum_{l=0}^{DD-1} y_{il} = d_i \quad ; i = 1,2,...,n \tag{10}$$

$$w_{it} = \frac{\sum_{l=t-T}^{t-1} y_{il}}{d_i} \quad ; i = 2,3,...,n-1, \quad t = T,2T,...,mT \tag{11}$$

$$S_i, SR_k, R_k \geq 0, \quad X_{it}, y_{it} = \{0,1\}, \; \forall i,t,k \tag{12}$$

The objective function in equation (1) maximizes the net present value of the project cash flows. The constraint (2) maintains the zero-lag finish-to-start precedence relations among the activities. The project due date, *DD*, is guaranteed in equation (3). Constraints (4) and (5) correspond to the recruiting and expulsion time points of the resources. Equation (6) ensures that the provided resource units are sufficient to implement the schedule. Equation (7) states that every activity must be started only once. Equation (8) computes the starting time of activities based on variables X_{it}. Equation (9) and equation (10) are to signal when an activity is in progress. Equation (11) represents the portion of work completed during period [*t*-*T*, *t*) for each activity *i*. Sets of constraints (12) denote the domain of the variables.

Mohring (1984) proved that RIP is an NP-hard problem. The present problem can be converted to RIP by elimination of constraints 4, 5, 9, 10, 11, and reducing the non-linear function to the linear one minimizing the resource costs. As a result, the problem under study is also NP-hard.

3-Metaheuristics

In this section, we propose two widely employed metaheuristics, genetic algorithm and simulated annealing, for the problem. They are inspired from the GA proposed by Najafi and Niaki (2006) specially in terms of solution representation scheme. Both algorithms use similar solution representation and fitness function. The details of these algorithms and their solution representation are as follows:

3-1- Solution representation

We build a vector with *n* elements (genes), $(F_1, F_2, ..., F_n)$, each element is responsible for one activity such that F_i states the used floating time of activity *i* at the obtained schedule of the chromosome and is equal to the difference between S_i at the schedule and ES_i obtained by the critical path method. We note that $0 \le F_i \le TF_i$ where TF_i denotes the total floating time of activity *i* and is equal to the difference between the latest and the earliest starting time of that activity.
In addition, considering dependences between elements of a chromosome, we have:

$$\underset{\forall j \in P(i)}{Max}\{F_j + EF_j\} - ES_i \le F_i \le TF_i \tag{13}$$

Where EF_j is the earliest finish time of activity *j*.

In order to generate a chromosome, we use two approaches; the one generating the gene values sequentially from gene 1 to gene *n*, the other moving from gene *n* to gene 1 sequentially. The description of these approaches is as follows:

3-1-1- Forward approach for chromosome generation

In this approach, we generate gene 1, gene 2, gene 3, ..., and gene *n* of a chromosome consecutively by the following algorithm:

1. Let $i = 1$.

2. Gene 1 is an integer value triangularly distributed on the interval $[0, TF_1]$ according to function shown at equation (14)

$$f_x(x) = \frac{2 \times (TF_1 - x)}{(TF_1)^2} \quad , 0 \le x \le TF_1 \tag{14}$$

3. $i = i + 1$.

4. Gene *i* is an integer value triangularly distributed on the interval $[\underline{F}_i, TF_i]$ according to equation (15), where $\underline{F}_i$ can be obtained by equation (16),

$$f_x(x) = \frac{2 \times (TF_i - x)}{(TF_i - \underline{F}_i)^2} \quad , \underline{F}_i \le x \le TF_i \tag{15}$$

$$\underline{F}_i = \underset{\forall j \in P(i)}{Max}\{F_j + EF_j\} - ES_i \tag{16}$$

5. If $i = n$, stop, otherwise go to step 3.

3-1-2-Backward approach for chromosome generation

In this approach, we generate gene *n*, gene $n-1$, gene $n-2$, …, and gene 1, consecutively by the following algorithm :

1. Let $i = n$.

2. Gene *n* is an integer value triangularly distributed on the interval [0, TF_n] according to equation (17)

$$f_x(x) = \frac{2x}{(TF_n)^2} \quad , 0 \le x \le TF_n \tag{17}$$

3. $i = i - 1$.

4. Gene *i* is an integer value triangularly distributed on the interval $[0, \overline{F}_i]$ according to equation (18), where $\overline{F}_i$ can be obtained by equation (19).

$$f_x(x) = \frac{2x}{(\overline{F}_i)^2} \quad , 0 \le x \le \overline{F}_i \tag{18}$$

$$\overline{F}_i = \underset{\forall j \in Suc(i)}{Min} \{F_j + ES_j\} - d_i - ES_i \quad (19)$$

5. If $i = 1$, stop, otherwise go to step 3.

Evaluating the fitness of chromosomes is the next step. In doing so, the schedule of each chromosome is firstly derived by setting the starting times of activity i to $S_i = F_i + ES_i$. Then, accordingly, the resource profile of each schedule is decoded by calculating the required level of resources through the schedule. Subsequently, R_k, SR_k, and FR_k are obtained. Finally, the fitness of each solution is acquired by equation (1).

3-2-Genetic algorithm

We propose, here, a GA as a powerful search algorithm for the problem under study. At first, a random population of chromosomes is generated by the procedure described in section 3.1. In each iteration of GA, a fixed number of best chromosomes are copied to the next generation and the rest are filled with chromosomes created using a crossover, mutation and local search operator. The algorithm stops when a specific number of generations is produced. The crossover and mutation operators are defined in the following way.

3-2-1- Crossover operator

This operator depends on two definitions below.

Definition 1. Backward floating time of activity i F_i^-: the difference between the starting time of activity i and the possible earliest starting time of activity i, both determined at the same schedule which is calculated by equation (20).

$$F_i^- = F_i + ES_i - \underset{\forall j \in P(i)}{Max} \{F_j + EF_j\} \quad (20)$$

Definition 2. Forward floating time of activity i, F_i^+: the difference between the possible latest starting time of activity i and the finishing time of activity i which is calculated by equation (21).

$$F_i^+ = \underset{\forall j \in Suc(i)}{Min} \{F_j + ES_j\} - \left(F_i + ES_i + d_i\right) \quad (21)$$

Now, two chromosomes (parents) $P1$ and $P2$ are selected to create two children, $CH1$ and $CH2$ by the following algorithm in which superscripts define the schedules under investigation:

1) Generate a random integer value r from the interval [2, $n-1$].
2) Generate a random number q from the interval [0,1], if $q \geq 0.5$ go to step 3, otherwise go to step 4.
3) Let $F_j^{CH1} = F_j^{P1}$ and $F_j^{CH2} = F_j^{P2}$ for $j = 1,2,...,n$, then change the value of genes for $j = r+1,...,n$ by the use of the following equations:

$$\begin{cases} F_j^{CH1} = F_j^{CH1} - F_j^{-CH1} & if \ \left(F_j^{+P2} \ \& \ F_j^{-P2} > 0\right) \ or \ \left(F_j^{+P2} = F_j^{-P2}\right) \\ F_j^{CH1} = F_j^{CH1} - F_j^{-CH1} + \left(\dfrac{F_j^{-P2}}{F_j^{+P2} + F_j^{-P2}}\right) \times \left(F_j^{+CH1} + F_j^{-CH1}\right) & O.W \end{cases} \quad (22)$$

$$\begin{cases} F_j^{CH2} = F_j^{CH2} - F_j^{-CH2} & if \ \left(F_j^{+P1} \ \& \ F_j^{-P1} > 0\right) \ or \ \left(F_j^{+P1} = F_j^{-P1}\right) \\ F_j^{CH2} = F_j^{CH2} - F_j^{-CH2} + \left(\dfrac{F_j^{-P1}}{F_j^{+P1} + F_j^{-P1}}\right) \times \left(F_j^{+CH2} + F_j^{-CH2}\right) & O.W \end{cases} \quad (23)$$

4) Let $F_j^{CH1} = F_j^{P1}$ and $F_j^{CH2} = F_j^{P2}$ for $j = 1,2,...,n$, then change the value of genes for $j = r+1,...,n$ by the use of the following equations:

$$\begin{cases} F_j^{CH1} = F_j^{CH1} + F_j^{+CH1} & if \ \left(F_j^{+P2} \ \& \ F_j^{-P2} > 0\right) \ or \ \left(F_j^{+P2} = F_j^{-P2}\right) \\ F_j^{CH1} = F_j^{CH1} + F_j^{+CH1} - \left(\dfrac{F_j^{+P2}}{F_j^{+P2} + F_j^{-P2}}\right) \times \left(F_j^{+CH1} + F_j^{-CH1}\right) & O.W \end{cases} \quad (24)$$

$$\begin{cases} F_j^{CH2} = F_j^{CH2} + F_j^{+CH2} & if \ \left(F_j^{+P1} \ \& \ F_j^{-P1} > 0\right) \ or \ \left(F_j^{+P1} = F_j^{-P1}\right) \\ F_j^{CH2} = F_j^{CH2} + F_j^{+CH2} - \left(\dfrac{F_j^{+P1}}{F_j^{+P1} + F_j^{-P1}}\right) \times \left(F_j^{+CH2} + F_j^{-CH2}\right) & O.W \end{cases} \quad (25)$$

3-2-2-Mutation operator

Let *P* be the chromosome selected for mutation. The steps below describe the mutation process:

1) Generate two uniformly distributed integers on the interval $[1, n-1]$. We call the smaller and larger one as r_1 and r_2, respectively. If $r_1 = r_2$ or $r_1 = r_2 - 1$, then copy chromosome *P* to the next population, absolutely and finish the algorithm, otherwise go to step 2.
2) Generate a uniformly distributed random value *q* between zero and one. If $q \geq 0.5$, go to step 3, otherwise go to step 4.
3) Change randomly gene *i*, $i = r_1 + 1,..., r_2 - 1$ to an integer value, triangularly distributed according to equation (26) (from gene $r_1 + 1$ to gene $r_2 - 1$, sequentially)

$$f_x(x) = \frac{2 \times \left(F_i^P + F_i^{+P} - x\right)}{\left(F_i^{+P} + F_i^{-P}\right)^2}, \quad F_i^P + F_i^{-P} \leq x \leq F_i^P + F_i^{+P} \quad (26)$$

4) Change randomly gene *i*, $i = r_1 + 1,..., r_2 - 1$ to an integer value, triangularly distributed according to Eq. (27) (from gene $r_2 - 1$ to gene $r_1 + 1$, sequentially)

$$f_x(x) = \frac{2 \times \left(x - F_i^P + F_i^{-P}\right)}{\left(F_i^{+P} + F_i^{-P}\right)^2}, \quad F_i^P + F_i^{-P} \leq x \leq F_i^P + F_i^{+P} \quad (27)$$

3-2-3-Local search operator

After a chromosome gets out of mutation operation, this local search operator is performed to find better solutions in the neighborhood of that chromosome. The operator chooses each gene one by one and changes its corresponding floating time. If the change results in a better fitness, that will be fixed, otherwise other changes will be examined. The operator works as follows.

Let i = 2;
  While $i < n$
    If Forward float (i) > 0
      $Fi = Fi + 1$;
      Leave the rest unchanged;
      Calculate new objective function Z^*;
      If Z^* is a better objective function, fix *Fi*;
      Else, Leave Fi unchanged;

Else if Backward Float (i) > 0

$Fi = Fi - 1$;

Leave the rest unchanged;

Calculate new objective function Z^*;

If Z^* is a better objective function, Fix Fi;

Else, Leave Fi unchanged;

Else

$i = i + 1$;

End

The parameters of GA have high influence on its performance and a good calibration of parameters reinforces the algorithm efficiency. The population size, crossover probability, mutation probability, and the number of generations have been determined by researches as some of the most important parameters of GA. In the following sub-section, we describe a statistical method for parameter calibration of this GA.

3-2-4-Parameter calibration of the GA

Having considered the parameters of GA as input variables and its results as output, the response surface methodology abbreviated to RSM (Myers and Montgomery, 1995) is applied for calibration. Here, the population size, crossover probability, mutation probability, and the number of generations are considered as input variables of RSM. In addition, the fitness function as accuracy index and the CPU time consumed as the time index of the algorithm are considered as outputs. To experiment RSM, we define a lower, a high, and a middle point for each variable or parameter according to table 1. In this table, n shows the number of project activities. In the RSM application, the value of parameters Pop, P_{cr}, P_{mu}, and G are coded to -1, 0, and 1 for their low, middle, and high levels, respectively.

Table 1. Search range and the levels of the input variables for the GA

Parameter	Variable	Range	low	middle	High
Population size (Pop)	X_1	n-8n	n	4n	8n
Crossover probability (P_{cr})	X_2	0.6-0.9	0.6	0.75	0.9
Mutation probability (P_{mu})	X_3	0.0-0.1	0.0	0.05	0.1
Number of generations (G)	X_4	50-300	50	175	*300*

The RSM is an optimization technique to find the optimum level of a known response function obtained from a set of variables. Since such function is unknown, it makes sense to use a design that provides equal precision of estimation of the response function. The central composite face-centered (CCF) is an efficient design used for fitting a second order response function. In this research, a 2^{4-1} fractional factorial CCF design with 4 central points and 8 axial points, (±1,0,0,0), (0,±1,0,0), (0,0,±1,0) and (0,0,0,±1), is selected to estimate the second order function. The CCF design is shown in table 2.

Then, a set of 45 test problems generated by PROGEN including problems with 20, 30 and 40 activities and 3, 4 and 5 resources are taken into account for experiments. The test problems are solved based on different levels of CCF defined in Table 2. The first response (Y1) is the accuracy performance index of GA defined as the ratio of the solution of GA to the maximum solution detected among 20 levels. The data in column Y1 is the mean of accuracy index for 45 test problems. The higher the value of Y1 is the better performance of the GA will occur. The second response (Y2) is the CPU time performance of GA defined as the ratio of the minimum time obtained among 20 levels to the time consumed by GA for each level. The data in column Y2 is the mean of time index for 45 test problems. The higher the value of Y2 is the better the performance of GA will occur.

Table 2. Result of the RSM experiments on the GA

Runs	Input variables				Response variables	
	X_1	X_2	X_3	X_4	Y_1	Y_2
1	0	0	0	0	0.0696	0.9633
2	0	0	0	0	0.0698	0.9607
3	0	0	0	0	0.0700	0.9702
4	0	0	0	0	0.0699	0.9671
5	1	0	0	0	0.0413	0.9849
6	-1	0	0	0	0.2231	0.9452
7	0	1	0	0	0.0645	0.9712
8	0	-1	0	0	0.0754	0.9181
9	0	0	1	0	0.0618	0.9451
10	0	0	-1	0	0.0797	0.9682
11	0	0	0	1	0.0387	0.9836
12	0	0	0	-1	0.2773	0.9362
13	1	-1	1	-1	0.1600	0.8770
14	-1	1	1	-1	0.7045	0.8268
15	1	1	-1	-1	0.1717	0.9659
16	-1	-1	-1	-1	1.0000	0.8871
17	1	1	1	1	0.0188	0.8798
18	-1	1	-1	1	0.1317	0.9645
19	-1	-1	1	1	0.1193	0.9238
20	1	-1	-1	1	0.0288	0.9315

The results obtained from the analysis of variance of CCF are then used to estimate the second order functions for each response. The fitted responses are shown in equations (28) and (29).

$$Y1 = 0.969 + 0.019X_1 - 0.026X_2 + 0.007X_3 + 0.009X_4 - 0.012X_1^2 - 0.015X_2^2 - 0.027X_3^2 - 0.007X_4^2 - 0.026X_2X_3 - 0.005X_2X_4 + 0.007X_3X_4 \quad (28)$$

$$Y2 = 0.058 - 0.198X_1 - 0.035X_2 - 0.029X_3 - 0.176X_4 + 0.108X_1^2 + 0.021X_2^2 + 0.020X_3^2 + 0.082X_4^2 + 0.146X_2X_3 + 0.036X_2X_4 + 0.036X_3X_4 \quad (29)$$

The final goal of RSM is to find a desired level of the GA parameter values such that both the accuracy index and the CPU time index of the GA are simultaneously optimized. In fact, we tackle a bi-objective decision-making problem with conflicting objectives. We use the weighted additive fuzzy goal programming (Tiwari et al., 1987) which converts the model into a single objective function defined as the weighted sum of achievement degrees of the goals with respect to their target values. The method uses a single utility function to show the overall preference index, which may be any aggregated function of all achieved values of goals for each feasible solution. To solve this problem, we first need to obtain the payoff table of the positive ideal solution (PIS) as presented in table 3.

Table 3. Payoff table of PIS (in GA)

	Y_1	Y_2
Max Y_1	1.009336	0.875
Max Y_2	-0.5534	0.766

The membership functions of these two objectives can be obtained as follows:

$$\mu_{Y_1} = \begin{cases} 0, & Y_1 < 0.875 \\ \dfrac{Y_1 - 0.875}{1.009336 - 0.875}, & 0.875 \le Y_1 \le 1.009336 \\ 1, & Y_1 > 1.009336 \end{cases} \quad (30)$$

$$\mu_{Y} = \begin{cases} 0, & Y_2 < -0.5534 \\ \dfrac{Y_2 + 0.5534}{0.766 + 0.5534}, & -0.5534 \le Y_2 \le 0.766 \\ 1, & Y_2 > 0.766 \end{cases} \quad (31)$$

Then, the model becomes:

$$\begin{aligned} &Max \quad Z = \sum_{j=1}^{2} w_j \alpha_j \\ &Subject\ to \quad \mu_{Y_j} \ge \alpha_j; \quad j = 1, 2 \\ &\qquad -1 \le X_i \le 1; \quad i = 1, 2, 3, 4 \\ &\qquad \alpha_j \in [0,1]; \quad j = 1, 2 \end{aligned} \quad (32)$$

Where α_j and w_j denote the achievement degree and the weight of the j^{th} goal, respectively. Since the accuracy index is more important than the CPU time index, we chose $w_1 = 0.75$ and $w_2 = 0.25$. Finally, the optimum values of the GA parameters are obtained and presented in Table 4.

Table 4. Optimum value of input variables for the GA

Parameter	**Optimum value**
Population size (Pop)	8n
Crossover probability (P_{cr})	0.6
Mutation probability (P_{mu})	0.08
Number of generations (G)	300

3-3-Simulated annealing algorithm

The simulated annealing as a well-known local search metaheuristic starts its search on an initial solution and moves gradually toward the better solutions using a neighborhood search. Naturally, SA starts in an initial temperature (T_0) and finds a number of neighborhoods in that temperature. When a neighbor is produced, if its objective value is better than that of the current solution, it will take the position of the current solution. Even the worse neighbors have a chance to be kept alive during the search with a small probability, of course. Then, the temperature is reduced with a function named cooling scheme and again a number of neighbors are visited. As regards the problem under consideration, at first, a population of solutions is generated using two approaches previously explained in sections 3.1.1 and 3.1.2 and the best solution of the population is chosen as the initial solution of the algorithm. Then, neighbors are generated as follows:

3-3-1- Creating neighborhood and replacing

1) Generate an integer value *g*, uniformly distributed on the interval [1, *n*].
2) According to the value of *g*, decide as follows:
   - If $g = 1$, then replace elements 1 and 2

- If $g = n$, then replace elements n and $n-1$
- Otherwise, generate a random value r, uniformly distributed on the interval [0, 1]. If $r \leq 0.5$, replace elements g and $g-1$, otherwise replace elements g and $g+1$.

3) Test the feasibility of the solution. If the solution is feasible, go to step 4, otherwise restore the changes and go to step 1.
4) Calculate the objective value of the neighbor called FF_n, Let $\varepsilon = FF_n - FF_c$ where FF_c represents the objective value of the current solution.
5) Generate a random variable q, uniformly distributed between zero and one, if $\varepsilon \geq 0$ or $q \leq e^{(\varepsilon/T)}$, replace the current solution by the neighbor, otherwise reject the neighbor and hold the current solution. (T represents the temperature at each stage)

In each temperature, a specific number of neighbors are produced and then the temperature reduces with cooling function below. Note that T_i denotes the i^{th} temperature and $0 \leq \alpha \leq 1$ is a constant named cooling rate which is highly influential of the quality of SA's results.

$$T_i = \alpha T_{i-1} \tag{33}$$

The algorithm stops when the temperature reaches to a previously defined final temperature.

3-3-2-Parameter calibration of SA

Similar to the GA, the RSM is applied to tune the SA parameters. Four parameters; initial temperature, cooling rate, number of repetition at each temperature, and final temperature are considered as input variables. Table 5 presents the search range and the levels of the input variables. In the table, n shows the number of activities of project.

Table 5. Search range and the levels of the input variables for the SA algorithm

Parameter	Variable	Range	Low	Middle	High
Initial temperature (T_0)	X1	50-150	50	100	150
Cooling rate (α)	X2	0.8-0.99	0.8	0.895	0.99
number of repetition at each temperature (N)	X3	n-8n	n	4n	8n
Final temperature (T_f)	X4	0.0001-10	0.0001	5.00005	10

Similarly, a 2^{4-1} fractional factorial CCF design with four central points and 8 axial points are used to do the analysis of variance and estimate the second order response functions. Y_1 and Y_2 are the accuracy index and CPU time index, respectively. Table 6 represents the input variable levels and their corresponding responses tested on 45 previously introduced problems in section 3.2.3.

Table 6. Result of the RSM experiments on the SA algorithm

Runs	Input variables				Response variables	
	X_1	X_2	X_3	X_4	Y_1	Y_2
1	0	0	0	0	0.2746	0.9364
2	0	0	0	0	0.2707	0.9353
3	0	0	0	0	0.2675	0.9360
4	0	0	0	0	0.2685	0.9404
5	1	0	0	0	0.3518	0.9408
6	-1	0	0	0	0.0722	0.9346
7	0	1	0	0	0.1739	0.9318
8	0	-1	0	0	0.6953	0.9311
9	0	0	1	0	0.0322	0.9410
10	0	0	-1	0	0.4716	0.9302
11	0	0	0	1	0.2553	0.9298
12	0	0	0	-1	0.3528	0.9322
13	1	-1	1	-1	0.1984	0.9371
14	-1	1	1	-1	0.0044	0.9332
15	1	1	-1	-1	0.4890	0.9384
16	-1	-1	-1	-1	0.4483	0.9256
17	1	1	1	1	0.0199	0.9349
18	-1	1	-1	1	0.0823	0.9332
19	-1	-1	1	1	0.0266	0.9377
20	1	-1	-1	1	1.0000	0.9332

Using the fuzzy goal programming for solving the functions estimated by the CCF above, the optimal values of the parameters are defined in table 7.

Table 7. Optimum value of input variables for the SA algorithm

Parameter	**Optimum value**
Initial temperature (T_0)	$99.53558505 \approx 100$
Cooling rate (α)	0.99
number of repetition at each temperature (N)	$3.2440612n \approx 3n$
Final temperature (T_f)	0.0001

4-Computational experiments

In this section, the results obtained from examining the proposed GA and SA on some test problems are reported. Collections of problems generated by PROGEN with 20, 30, 40, 80, 100, and 120 activities are considered. For the set of 20, 30, and 40 activities, a collection of 3, 4, and 5 resources, each including 10 problems is examined. For the set of 80, 100, and 120 activities, a collection of 5 test problems with 4 resources are examined. Both algorithms, GA and SA, were programmed in Matlab software and then tested on the 105 produced test problems. Moreover, the mathematical formulations of these problems are also programmed in LINGO software to compare its results with the proposed algorithms. Throughout the running of LINGO, the CPU time is limited to 1800 seconds, i.e. if LINGO could not solve a problem at this time, the process is terminated. The experiments were performed on a PC with a Pentium 2000 core2 Duo processor and 3000MB RAM. Table 8 shows the comparison results of these two algorithms and also the results of solutions found by LINGO. In this table, seven comparison criteria are utilized. The symbols of the table are defined as follows:

: the number of instances for which the algorithm is able to found a solution before 1800 seconds
#b : the number of instances for which the algorithm found a solution equal to the best solution among three ones
AAD : the average absolute deviation from the best solution known
MAD : the maximal absolute deviation from the best solution known
ARD% : the average relative deviation from the best solution known
MRD% : the maximal relative deviation from the best solution known
CPU : the average computational time of the algorithm.

Table 8. Results of the experiments for the problem

No. of activities	No. of resources	No. of problems	#			#b			AAD		MAD		ARD%		MRD%		CPU time (s)		
			LINGO	GA	SA	LINGO	GA	SA	GA	SA	GA	SA	GA	SA	GA	SA	LINGO	GA	SA
20	3	10	10	10	10	10	0	0	25	53	42	100	8	18	14	31	15	14	26
	4	10	10	10	10	10	0	0	22	55	43	89	9	19	22	32	32	15	32
	5	10	10	10	10	10	0	0	22	56	32	75	5	14	8	22	13	16	32
30	3	10	9	10	10	9	1	0	51	110	92	167	21	42	47	64	603	36	54
	4	10	8	10	10	8	2	0	32	130	94	196	8	33	19	54	292	34	57
	5	10	9	10	10	9	1	0	39	137	63	196	8	24	13	43	256	40	59
40	3	10	5	10	10	5	5	0	31	161	90	239	7	32	18	50	842	73	83
	4	10	2	10	10	2	8	0	7	121	76	185	1	20	16	32	310	72	104
	5	10	1	10	10	1	9	0	2	122	22	174	0.4	18	3	27	450	84	103
80	4	5	0	5	5	0	5	0	0	966	0	2003	0	52	0	134	-	233	421
100	4	5	0	5	5	0	5	0	0	770	0	1019	0	28	0	35	-	1017	570
120	4	5	0	5	5	0	5	0	0	1450	0	2124	0	33	0	42	-	1520	980

5-Discussion and findings

In this section the reported results are analyzed and the findings are discussed in terms of seven criteria as follows.

#-based results: according to table 8, LINGO is not able to find a solution within 1800 seconds for 4 out of 30 problems with 30 activities, 22 out of 30 problems with 40 activities, and all of 80, 100, and 120 activities while the other algorithms, GA and SA, are able to find solutions for all problems in reasonable time. These results show that the proposed metaheuristic algorithms are promising in tackling the RIP with progress payment problems since in real world large scale problems where no exact solution is available, having access to near optimal solutions is favored.

#b-based results: according to table 8, LINGO has expectedly found the best solution among other algorithms where a solution exists. In other cases, i.e. where LINGO has no solution for the problem, the GA has reached the best ones. These results demonstrate that the performance of GA is more reliable than SA since the results of GA outperform those of SA in all of 105 problems.

AAD-based results: according to table 8, the average of ADD for GA and SA in 20-activity problems is 23 and 54, respectively, in 30-activity problems are 41 and 126, respectively, and in 40-activity problems are 13 and 135, respectively which substantiates previous results displaying better performance of GA. The AAD of 13 in the last set of activities for GA is a little strange, because when the size of the problems increases, it is expected that the accuracy of a metaheuristic based algorithm like GA decreases. The only reason for such outcome is the limited access to near optimal solutions in this set of problems as LINGO has found such solution for 8 out of 30 problems. In terms of large scale problems with 80, 100, and 120 activities, as there are no solutions by LINGO and the GA has found better results in all of test problems, the ADDs of GA are zero.

MAD-based results: according to table 8, these results also support higher acceptance of GA's results. The maximum absolute deviation for GA among all 105 problems is 94 while that is 2124 for SA. In addition, the average of MAD for GA and SA are 46 and 547, respectively denoting better efficiency of GA.

ARD%-based results: according to table 8, as expected, GA shows better results in this issue where the average of ARD% for SA (28) is more that four times of that for GA (6).

MRD%-based results: according to table 8, the MRD% for GA is smaller than that of SA in all sets of the problems confirming the better performance of GA.

CPU time-based results: according to table 8, the average of CPU time used by three methods in all 105 problems is 313, 263, and 210 seconds for LINGO, GA, and SA, respectively. This comparison demonstrates that, the more complex the problems, the more CPU time used by algorithms, nevertheless LINGO could not tackle the large scale problems in reasonable time and its CUP time used grows exponentially while proposed algorithms do not show such behavior. In addition, the CPU time used by GA is lesser than that of SA in smaller problems (30 to 80 activities) while in larger problems (100 and 120 activities) the SA consumes shorter time to reach solutions.

In a nutshell, although LINGO finds optimal solutions in small problems, it is not applicable in coping with large problems while the proposed algorithms can solve the large problems in reasonable time. In addition, GA outperforms SA in regards to the accuracy of solutions in all of test problems. Although the GA solves smaller problems in shorter time, the SA finds a solution for larger problems sooner.

6-Conclusions and recommendations for future research

In this research, a class of project scheduling problems, called resource investment problem was considered. While, the RIP had been previously investigated under maximization of NPV where its payments occur at some events of the project, here the progress payment model was chosen to be combined with RIP. The mathematical formulation of the model was defined and due to NP-hardness of the problem, metaheuristic algorithms were taken into account to solve the model. Two well-known metaheuristics, SA and GA, were designed. In order to improve the efficiency of the proposed GA and SA, their parameters were statistically tuned using response surface methodology. Then, the tuned algorithms were examined on 105 test problems of small, medium, and large scales. The results of the applications of the proposed methodologies on these problems and the comparison of the results with solutions obtained from solving the mathematical formulation of the problems by LINGO software showed that the performance of GA not only is reliable when compared to LINGO results, is also highly better than SA algorithm.
Some of potential extensions of the model are considering precedence relations with minimal and maximal time lag, multi-mode execution of activities, and permitting the earliness or tardiness deviation from the due date of the project with a bonus and penalty policy.

References

Afshar-Nadjafi, B. (2014). Multi-mode resource availability cost problem with recruitment and release dates for resources. *Applied Mathematical Modelling*, *38*(21-22), 5347-5355.

Ahuja, H. N. (1976). *Construction performance control by networks*. New York; Toronto: Wiley.

Demeulemeester, E. (1995). Minimizing resource availability costs in time-limited project networks. *Management Science*, *41*(10), 1590-1598.

Doersch, R. H., & Patterson, J. H. (1977). Scheduling a project to maximize its present value: A zero-one programming approach. *Management Science*, *23*(8), 882-889.

Drexl, A., & Kimms, A. (2001). Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society*, *52*(3), 340-351.

Elmaghraby, S. E., & Herroelen, W. S. (1990). The scheduling of activities to maximize the net present value of projects. *European Journal of Operational Research*, *49*(1), 35-49.

Etgar, R. (1999). Scheduling project activities to maximize the net present value the case of linear time-dependent cash flows. *International Journal of Production Research*, *37*(2), 329-339.

Etgar, R., Shtub, A., & LeBlanc, L. J. (1997). Scheduling projects to maximize net present value—the case of time-dependent, contingent cash flows. *European Journal of Operational Research*, *96*(1), 90-96.

Etgar, R., & Shtub, A. (1997). A branch and bound algorithm for scheduling projects to maximize net present value: the case of time dependent, contingent cash flows. *International Journal of Production Research*, *35*(12), 3367-3378.

Grinold, R. C. (1972). The payment scheduling problem. *Naval Research Logistics Quarterly*, *19*(1), 123-136.

Herroelen, W. S., & Gallens, E. (1993). Computational experience with an optimal procedure for the scheduling of activities to maximize the net present value of projects. *European Journal of Operational Research*, *65*(2), 274-277.

Javanmard, S., Afshar-Nadjafi, B., & Niaki, S. T. A. (2017). Preemptive multi-skilled resource investment project scheduling problem: Mathematical modelling and solution approaches. *Computers & Chemical Engineering*, *96*, 55-68.

Kazaz, B., & Sepil, C. (1996). Project scheduling with discounted cash flows and progress payments. *Journal of the Operational Research Society*, *47*(10), 1262-1272.

Möhring, R. H. (1984). Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Operations Research*, *32*(1), 89-120.

Myers, R. H., Montgomery, D. C., Vining, G. G., Borror, C. M., & Kowalski, S. M. (2004). Response surface methodology: a retrospective and literature survey. *Journal of quality technology*, *36*(1), 53-77.

Najafi, A. A., & Azimi, F. (2009). A priority rule-based heuristic for resource investment project scheduling problem with discounted cash flows and tardiness penalties. *Mathematical Problems in Engineering*, *2009*.

Najafi, A. A., & Niaki, S. T. A. (2006). A genetic algorithm for resource investment problem with discounted cash flows. *Applied Mathematics and Computation*, *183*(2), 1057-1070.

Najafi, A. A., Niaki, S. T. A., & Shahsavar, M. (2009). A parameter-tuned genetic algorithm for the resource investment problem with discounted cash flows and generalized precedence relations. *Computers & Operations Research*, *36*(11), 2994-3001.

Nübel, H. (1999). *A branch and bound procedure for the resource investment problem subject to temporal constraints*. Inst. für Wirtschaftstheorie und Operations-Research.

Nübel, H. (2001). The resource renting problem subject to temporal constraints. *OR-Spektrum*, *23*(3), 359-381.

Qi, J. J., Liu, Y. J., Jiang, P., & Guo, B. (2015). Schedule generation scheme for solving multi-mode resource availability cost problem by modified particle swarm optimization. *Journal of Scheduling*, *18*(3), 285-298.

Ranjbar, M., Kianfar, F., & Shadrokh, S. (2008). Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. *Applied Mathematics and Computation*, *196*(2), 879-888.

Rodrigues, S. B., & Yamashita, D. S. (2010). An exact algorithm for minimizing resource availability costs in project scheduling. *European Journal of Operational Research*, *206*(3), 562-568.

Russell, A. H. (1970). Cash flows in networks. *Management Science*, *16*(5), 357-373.

Sabzehparvar, M., SEYED, H. S., & Nouri, S. (2008). A mathematical model for the multi-mode resource investment problem.

Sepil, C., & Ortac, N. (1997). Performance of the heuristic procedures for constrained projects with progress payments. *Journal of the Operational Research Society*, *48*(11), 1123-1130.

Shadrokh, S., & Kianfar, F. (2007). A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. *European Journal of Operational Research*, *181*(1), 86-101.

Shahsavar, M., Najafi, A. A., & Niaki, S. T. A. (2011). Statistical design of genetic algorithms for combinatorial optimization problems. *Mathematical Problems in Engineering*, *2011*.

Smith-Daniels, D. E., & Aquilano, N. J. (1987). Using a late-start resource-constrained project schedule to improve project net present value. *Decision Sciences*, *18*(4), 617-630.

Tiwari, R. N., Dharmar, S., & Rao, J. R. (1987). Fuzzy goal programming—an additive model. *Fuzzy sets and systems*, *24*(1), 27-34.

Vanhoucke, M., Demeulemeester, E., & Herroelen, W. (1999). Scheduling projects with linear time-dependent cash flows to maximize the net present value.

Yamashita, D. S., Armentano, V. A., & Laguna, M. (2006). Scatter search for project scheduling with resource availability cost. *European Journal of Operational Research*, *169*(2), 623-637.

Zimmermann, J., & Engelhardt, H. (1998). Lower bounds and exact algorithms for resource leveling problems. *Report WIOR-517, University Karlsruhe*.

Zoraghi, N., Shahsavar, A., Abbasi, B., & Van Peteghem, V. (2017). Multi-mode resource-constrained project scheduling problem with material ordering under bonus–penalty policies. *Top*, *25*(1), 49-79.