KEYWORDS genetic algorithm project scheduling parameter control self-adaptive

DOI NUMBER: 10.19255/JMPM01105

A TWO-PART SELF-ADAPTIVE TECHNIQUE IN GENETIC ALGORITHMS for Project Scheduling Problems

Aria Shahsavar
Qazvin Islamic Azad University
m.shahsavar@ymail.com

Seyed Taghi Akhavan Niaki
Sharif University of Technology
niaki@sharif.edu

Amir Abbas Najafi
K.N. Toosi University of Technology
aanajafi@kntu.ac.ir

ABSTRACT

The present paper introduces a novel two-part self-adaptive technique in designing the genetic algorithm for project scheduling problems. One part of the algorithm includes a self-adaptive mechanism for genetic operators like crossover and mutation. The second part contains another self-adaptive mechanism for genetic parameters such as crossover probability. The parts come in turn repeatedly within a loop feeding each other with the information regarding the performance of operators or parameters. The capability of the method is tested and confirmed in comparison to metaheuristic and exact algorithms based on well-known benchmarks.

1. Introduction

The project scheduling problems *(PSP)* seek the most favorable schedule for a set of activities linked together with precedence relations in such a way that one or some goals are achieved while no violation of resource limits occurs **[1]**. Generally, network diagrams with activities and precedence relations represented by nodes and arcs are utilized to depict PSPs. The PSPs involve a wide variety of models, of which 1) resource-constrained project scheduling problem, 2) resource investment problem, and 3) resource leveling problem are known as the basic models upon which other models are created. According to **[2]**, PSPs are of the most demanding problems in operations research for which a large number of heuristics have been proposed. Among a multitude of heuristics applied on PSPs, the genetic algorithms *(GA)* have been one of the most successful and favorable methods. For a review of GAs devised for these general classes of PSPs, one can refer to **[3]** - **[17]**.

The GA inspires its soul by natural selection and survival of the fittest **[18]**. The algorithm goes through a population of solutions and evolves stepwise by joining the features of solutions. In each step, the current solutions are selected as parents to be recombined and mutated in hopes of discovering children with superior characteristics. Such operations generally are conducted by means of genetic operators such as Parent Selection, Crossover, and Mutation. To maintain the randomness of the search, the operators are carried out probabilistically during the GA by some random parameters such as Crossover Probability and Mutation Probability. As different alternatives for each component *(operator and parameter)* of GA is available in the literature, GA practitioners always face two questions. Which alternative of genetic operators should be chosen and on what level each parameter should be set to guarantee good results?

Many efforts have been put forwarded to answer the above questions. These efforts can be classified to parameter tuning and parameter control. Parameter tuning approaches seek a proper combination of alternatives for components of GA based on an analysis of results reported by the GA tested on a set of problems representative for the problem instances. Noticeably, the components are known before running the GA and remain constant during the run **[7]**. Of different parameter tuning approaches, the statistical methods are more favorable. For a review, please see **[19]** - **[30]**. According to **[7]**, although parameter tuning is suitable for a large number of situations, it may not lead to the best possible GA because the test problems not only are heterogeneous but also cannot be actually viewed as the representative for the real-world situations. In addition, due to the nature of some components that may perform well in early generations and some others that act better in later generations, parameter tuning may not work well. Therefore, the parameter control comes into picture as an alternative case in which the components are not fixed during the run of GA and the best combination is detected throughout the search. This case requires initial parameter values and suitable control strategies. References **[29]** and **[31]** classified the parameter control procedures into three classes of deterministic **[32]**, adaptive **[33]**, and self-adaptive. The third class, the self-adaptive parameter control procedure, encodes the components within each solution of the GA and discovers the best ones via a feedback-based control strategy. Reference **[34]** introduced an interesting self-adaptive mechanism for crossover operator in which one bit is appended to the end of each solution to show which alternative of the operator has created the solution. The bit helps the GA choose better alternatives with regard to their corresponding fitness. When, for example crossover operator, is to be performed, this bit signals the algorithm to employ the most adapted alternative for crossover. In doing so, the related bits of all the solutions are evaluated and the alternative with more repetition is known as the most adapted one. Reference **[35]** showed a mechanism to adapt the mutation probability in a model of parallel GA. Reference **[36]** includes two levels of GAs. While the meta-level evolves a population of parameters, the basic level operates on the best set of parameters obtained by the meta-level. Reference **[37]** introduced an approach for selecting parameters by establishing a competition among several subpopulations, where each subpopulation uses different sets of parameters. The additional processing time is given to the populations with better parameter sets. Some of the other adaptive GAs such as **[38]** - **[40]** are to adaptively control the diversity of population. Owing to the efficiency of self-adaptive techniques in the GA, they have been continually used over years in the literature (please see **[41]** - **[49]**).

Despite a multitude of GA-based efforts done on PSPs, a few of them have considered self-adapting GAs. Some of these works are **[7]**, **[50]** and **[51]**. Besides, most of self-adaptive GAs suggested for PSPs have focused only on one or a few characteristics of the GA to design an effective algorithm, whereas the GA has many components that impress its effectiveness and need more comprehensive attention. A robust GA cannot be devised unless best alternatives of genetic operators and parameters are revealed through a process examining different combinations of alternatives. Nevertheless, there has been less attention so far in designing the GAs for PSPs that utilize parameter control techniques, especially for the situation in which selecting good operators as well as setting good values to the parameters is desired simultaneously. In this paper, we introduce a more comprehensive method in which most of the vital elements impressing the performance of the GA are first designed. Then, a two-part self-adaptive GA in which one part relates to the adaption of operators and the other part attributes to the adaption of parameters is applied to bring a more thorough approach about.

The paper is structured as follows: Section 2 describes the proposed methodology for the PSPs. Section 3 investigates the application of the methodology on two different PSP problems. Section 4 gives the computational results of the proposed methodology in comparison to other metaheuristics as well as exact procedures. Finally, the conclusion of the paper comes in Section 5.

2. The proposed methodology

The present study addresses a new method that combines previously-devised parameter control techniques to tackle the PSPs. The proposed GA consists of two parts coming in turn repeatedly within a loop and providing input data for each other. One part *(OAP: operator-adapting part)* concentrates on genetic operators and uses a self-adapting mechanism to gradually detect the best alternative for each operator. This self-adapting mechanism actually acts like a feedback system that records and returns the fitness of alternatives when implemented on a population of basic problem solutions. Then in the later generations, high quality alternatives are given more opportunity to be implemented. The other part *(PAP: parameter-adapting part)* focuses on parameters and functions in such a way that discovers the best levels of parameters leading to better performances. Similarly, this part has a self-adapting mechanism that feeds the reports of parameter effects back to the algorithm.

2.1 Initialization of the proposed GA

At the advent of this GA, a population of the basic problem solutions *(P1)* for the PSP, and a population of probabilistic parameter sets *(P2)* is randomly produced. Both populations are equal in size and each solution of P2 contains a different set of parameters which contributes to one specific solution of P1. Besides, different alternatives are devised for genetic operators and are given the same opportunity to be implemented in the first step of algorithm.

2.2 Adaption rate calculations

The adaption rate means the opportunity given to each alternative or parameter value to be employed and changes within the algorithm based on the influence they have had on the evolution of GA. The adaption rate of an alternative is calculated based on the fitness of the solutions created by that alternative. Moreover, the adaption rate of each parameter set is also calculated based on the fitness of the solutions created by that set.

2.3 Fixing the alternatives and parameters

In order to eliminate the effects of parameters on the function of operators and vice versa, the OAP is executed in the presence of fixed parameters and the PAP is performed in the presence of constant operators. The parameters fed to OAP and operators fed to PAP are fixed by the following way. The value of parameter *i* is set to the average of all values of that parameter through the entire P2. In addition, one alternative for each operator is fixed probabilistically based on the adaption rates of alternatives. Actually for each alternative, the higher the adaption rate, the higher the chance of being fixed.

2.4 Operator-adapting part (OAP)

In the OAP, the approach suggested in **[34]** that appends a bit to the end of every solution for each operator is used. For instance, suppose an extra bit is added to the end of solution for crossover operator and let "0" refers to one-point crossover and "1" refers to uniform crossover. If the one-point crossover can create better solutions, then more "0" is appeared in the crossover-related bit during the evolution of GA. Note that, a mechanism that gives greater rewards to better alternatives is needed. For this purpose, imagine A1% of the extra crossover-related column of the population contains "0" and A2% contains "1". Let f_1 and f_2 be the mean fitness of A1% and A2% of the population, respectively. Then, the adaption rate of the crossover alternative *i* is calculated as $F_i = \frac{f_i}{\sum_{i=1}^{2} f_i}$. When recombining, either the one-point alternative with probability of F_1 or uniform alternative with probability of F_2 is selected. Therefore, if the one-point crossover performs better during the algorithm process, the higher F_1 is anticipated. After recombination, if the one-point crossover was the processor, the corresponding bit would change to "0" if necessary. As the algorithm evolves, better alternatives create bigger part of the population and their adaption rates increase.

After some generations, the OAP stops and introduces the more adapted alternatives with higher adaption rates. Note again that, the parameter values are constant during all generations of OAP.

2.5 Parameter-adapting part (PAP)

The PAP acts similar to **[36]**, in which two levels of genetic algorithms are considered. The meta-level evolves a population of parameter sets *(P₂)*, while the basic-level operates on the problem solution population *(P₁)*. In the meta-level of GA, a solution is represented by characteristics of parameters. For example, when the crossover probability and mutation probability are the subject of PAP, a vector with two cells; each including a real value for either of crossover or mutation probabilities is devised. The meta-level GA possesses its own characteristics and operators to operate on P_2 in pursuit of the best possible values of parameters. A question that may arise here is that how the fitness of parameter sets is evaluated. As mentioned before, each solution in the meta-level of GA is contributed to a specific solution in the basic-level of GA. Therefore, when a generation of P_2 is produced, a run of the basic GA is also conducted. In running the basic GA, the constant adaptive operators introduced by OAP is used together with the different parameter sets of P_2. Since the operators are constant and parameters are different, different results of the basic-level can be strongly referred to the effects of parameters. As a result, the fitness of each solution in the basic-level is considered as the fitness of its corresponding parameter set. Next, relying on the known fitness of parameter sets, another generation of meta-level GA is produced and its fitness is calculated by running the basic GA accordingly. The process proceeds for a few numbers of generations. Finally, the parameters adapted on proper sets are fed to the OAP.

2.6 The general scheme of the proposed GA

The algorithm starts with OAP focusing only on the adaption of potential alternatives for operators. The OAP repeats its contents giving greater rewards to well-performed alternatives to produce some generations while recording the adaption rates of alternatives. Then the alternatives with higher adaption rates are introduced to the PAP. The PAP informed of adapted alternatives proceeds for a specific number of generations while recording the adaption rates of parameters. Finally, the PAP transmits the adapted parameters to the OAP. Now one loop finishes and the other starts. This process proceeds until the stopping criterion of the algorithm is met. The pseudo-code of the proposed GA is shown in **Figure 1** wherein *N* is an even positive integer that specifies the number of generations produced within both parts of the algorithm. In addition, $L \leq \frac{N}{2}$ shows the number of loops needed to terminate the algorithm.

Since operators have much fewer alternatives than parameters whose alternatives are selected within a continuous range, the algorithm needs more efforts in order to find the adapted values of the parameters. Therefore, in this GA the more loops are run, the more generations are assigned to self-adapting mechanism of parameters.

The flowchart of the proposed GA is also shown in **Figure 2**.

A striking point regarding the proposed method is that, it is applicable to any branch of PSPs because of its compatibility with any type of the solution representations known in the literature for the PSPs. Typically, the most prevalent representation styles used in PSPs can be classified into three types; first, an activity list representation **[6]**, second, the random key representation **[13]**, and third, the real schedule representation such as a vector of starting times **[9]**. Adding the extra bits to solutions employed in OAP and using a sequential GA like the one employed in the PAP are both possible in all of these solution representation kinds. Therefore, the method can be applied on every branch of PSPs.

3. Application of the methodology on PSP

The resource investment problem (RIP) and its extensions are a class of PSPs in which the resource availabilities are considered as decision variables of the model. In the following sub-sections, the proposed methodology is firstly applied on the basic RIP and then is applied on the RIP with discounted cash flows (RIPDCF) to provide more reliable results.

3.1 Application on RIP

The basic RIP aims at minimizing the renewable resource costs of the project **[52]**. Suppose a project in an AON network G=*(E,V)* in which V shows activities and E shows the precedence relations of activities. The project has n+1 activities where two dummy activities 0 and n represent the starting and finishing points of the project, respectively. The generalized precedence relations with minimal and maximal time lags link activities together where l_{ij} is the time lag between the starting times of activities *i* and *j*, $(i,j) \in G$. The RIP with generalized precedence relations is referred to RIP/max. Each activity *i* has a fixed duration d_i and requires r_{ik} units of resource *k, k=1, 2, ... , K* to be executed. Employing each unit of resource k imposes a cost CK on the project. In addition, the project has to be finished before deadline *T*. The objective of the model is finding a schedule for activities and a level of employment for resources so that the total costs of the project resources are minimized. The RIP/max *(RIP under generalized precedence relations with minimal and maximal time lags between activities)* can be formulated in the following way.

$$Min \quad \sum_{k=1}^{K} C_k R_k \tag{1}$$

Subject to:

$$\sum_{t=ES_i}^{LS_i} x_{it} = 1; \quad \forall i = 0,1,\dots,n; \tag{2}$$

$$\sum_{t=ES_n}^{LS_n} t x_{nt} \leq T; \tag{3}$$

$$\sum_{t=ES_i}^{LS_i} t x_{it} + l_{ij} \leq \sum_{t=ES_j}^{LS_j} t x_{jt}; \; \forall (i,j) \in G; \tag{4}$$

$$\sum_{i=0}^{n} \sum_{w=\max(t-d_i+1,ES_i)}^{\min(t,LS_i)} r_{ik} x_{iw}; \quad \forall t = 1,2,\dots,T; \forall k = 1,2,\dots,K; \tag{5}$$

$$x_{it} = \{0,1\}; \; R_k \geq 0; \forall i = 1,2,\dots,n; \; \forall k = 1,2,\dots,K; \; \forall t = 1,2,\dots,T; \tag{6}$$

In this formulation, x_{it} is a zero-one variable where $x_{it} = 1$ if the activity *i* starts at time period *t*, otherwise $x_{it} = 0$. R_k is a positive variable showing the employment level of resource *k*. The objective function *(1)* minimizes the resource costs of the

```
Initialize  $P_1, P_2, N, \& L$;
$l = 0$;
While  $l \leq L$
    • DO OAP:
      -  $i = 1$;
      -  $N_1 = \text{round}\left(\left(\frac{N}{2}\right)-\left(\frac{N}{2L}\right).l\right)$;
      -  Fix the parameter values;
      -  While  $i \leq N_1$
           ✓ Using the adaption rate of alternatives Do:
                o Parent selection
                o Crossover
                o Mutation
           ✓ Update adaption rate of alternatives
           ✓ $i = i + 1$;
      -  End
      -  Transmit the final adaption rates of operators to PAP
    • DO PAP:
      -  $j = 1$;
      -  $N_2 = N - N_1$;
      -  Fix the operator alternatives;
      -  While  $j \leq N_2$
           ✓ Do basic GA:
                o Parent selection
                o Crossover
                o Mutation
                o Do meta-level GA for parameters:
                     ➢ Parent selection
                     ➢ Crossover
                     ➢ Mutation
           ✓ $j = j + 1$;
      -  End
      -  Transmit the final adaption rates of parameters to OAP
    • $l = l + 1$;
End
```

FIGURE 1: *The pseudo-code of the proposed GA*

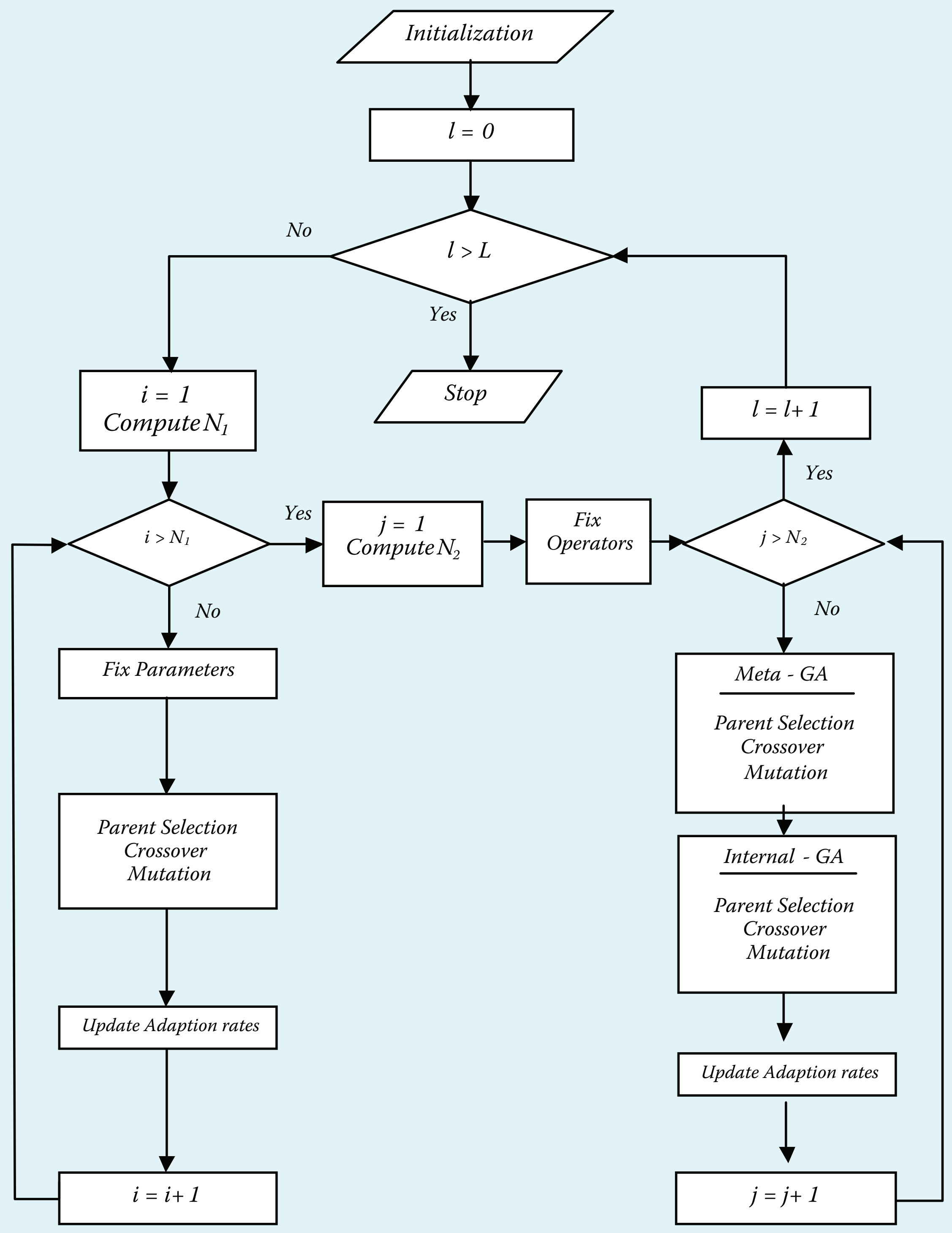


FIGURA 2: *The general framework of the proposed algorithm*

project. Constraint *(2)* forces every activity *i* to be started only once between its earliest and latest starting time, ES_i and LS_i respectively. Constraint *(3)* states that the project should be finished before deadline *T*. Constraint *(4)* shows generalized precedence relations. Constraint *(5)* shows resource constraints. Constraint *(6)* introduces the domain of variables.

To utilize the proposed methodology on the RIP/max, we borrow the solution representation of **[30]**. The solution is a vector including $n+1$ cells *($n+1$ shows the number of project activities)* such that the value of cell *i* indicates the starting time of activity *i*. In order to create a solution, the activities are selected randomly and their starting times are produced among their earliest and latest starting times at random. After determining the starting time of each activity, the earliest and latest starting times of its predecessors and successors are updated to prevent the solution from being infeasible. In **[30]**, different alternatives for Parent Selection, Crossover, and Mutation operators have also been designed. The alternatives designed in **[30]** are employed here to be the subject of the present self-adaptive GA. The description of designed alternatives for each component in **[30]** is as follows.

The tournament selection, which is a fitness-based parent selection, and the unlike selection that is not a fitness-based parent selection have been devised as parent selection alternatives. The one-point crossover and uniform crossover are the designed alternatives for recombination. In one-point crossover, two parents are cut in a middle point of the vector. The first part of the child is filled directly by cells of the left part of the first parent and the right part is filled by the use of the cells of the right part of the second parent as follows. At first, the earliest and latest starting times of activities with empty cells are updated based on the starting times of the left activities. Then for each cell *i*, if the value of the corresponding cell of the second parent is positioned between the updated ES_i and LS_i, it is fixed in the child, if it is smaller than the updated ES_i, the cell value is set to ES_i, otherwise it is set to LS_i. In uniform crossover, the starting time of each activity of child is chosen randomly among its corresponding cell of one parent. Notice that, when a cell value is fixed, the value of its successors and predecessors may be shifted forward or backward to prevent the child from being infeasible. In terms of mutation, two strategies are applied. In the first mutation strategy, a mutation probability is assigned to each child and then, either all or none of the cells of that child are mutated. In the second strategy, a mutation probability is assigned to each cell and then, that cell is mutated independent of others. In order to mutate one cell or one activity, its starting time is replaced with new value generated randomly among its corresponding earliest and latest starting time. Then, the starting times of its successors and predecessors are updated for guaranteeing the feasibility.

For the PAP, we choose crossover probability and mutation probability as the subject of self-adaption. Then, we design a two-cell vector in which cell 1 relates to the crossover probability and cell 2 relates to the mutation probability. The cell 1 is randomly valued from the intervals [0.6,1] and cell 2 within [0,0.3]. The self-adaptive mechanisms for both operators and parameters are described as follows.

a) OAP for RIP/max

We appended 3 bits at the end of each solution to show different alternatives of three operators i.e. parent selection, crossover, and mutation operator. The alternatives introduced before for each operator are given equal adaption rates in the first OAP. The GA is initialized according to Section 2.1 and then the OAP starts. During the OAP, the parameters fed by the PAP are fixed and the parents are selected by either tournament or unlike selection regarding to their adaption rates. They are then recombined by either one-point or uniform crossover and continue with the mutation. The adaptive mechanism relies on the fitness of solutions. The best objective function found during the whole run of GA is maintained. The fitness value of each solution is calculated based on division of the best objective function found to the objective function of that solution. Accordingly, the adaption rate of alternatives is computed via *(7)*.

$$F(j,i)=100f(j,i)\Big/\sum_{j=1}^{N_i}f(j,i) \quad (7)$$

In *(7)*, N_i is the number of alternatives of operator i and *f(j,i)* is the mean fitness of the solutions created by the j^{th} alternative of operator *i* in the entire of population. Then, *F(j,i)* calculates the adaption rate of the j^{th} alternative of operator *i*. It means that when one operator like *i* is on function, its j^{th} alternative is utilized with the probability of *F(j,i)*. This part terminates when a specific number of generations are created and then reports the adapted alternatives to the PAP.

b) PAP for RIP/max

In the PAP, the previously described two-level GA is applied. In the meta-level of GA, a solution in P_2 is a vector with two cells for crossover and mutation probabilities. The meta-level of GA selects solutions in P_2 by tournament selection and recombines them by uniform crossover in which each cell of parents has an equal chance to be present in the child. For mutation, a cell value of a child is randomly replaced by another random value drawn between its corre-

Group	No. of activities	No. of bench-marks	No. of resources	Average. of RDP1%
J10	10	90	1	0
		90	3	0.1
		90	5	0.5
J20	20	90	1	1.1
		90	3	1.8
		90	5	2
J30	30	90	1	1.5
		90	3	2.5
		90	5	4.4

TABLE 1: *Comparison results with exact solutions of RIP/max*

sponding range. The fitness of solutions in P_2 is calculated by running the basic level of GA wherein operators come from adapted ones in the OAP and parameters from P_2. After running the basic GA, the fitness of solutions in new P_1 is set to the corresponding solutions in P_2. Now, an iteration of the meta-level GA terminates and another one starts. By this way, not only the superior parameter sets are discovered but also the quality of the basic problem population improves. This part terminates after a few generations and feeds the adapted parameters to the OAP.

3.2 Application on RIPDCF

Reference **[14]** investigates the RIP in the presence of negative and positive cash flows with minimal and maximal time lags *(RIPDCF/max)* whose goal is to find a schedule as well as the employed levels of resources such that the net present value *(NPV)* of the project cash flows is maximized. The RIPDCF/max possesses some other characteristics in addition to those of the RIP/max. Some payments are received at some points of the project when a set of activities finishes. Furthermore, a cost including material and overhead costs is imposed on the project based on each activity. The mathematical formulation of RIPDCF/max has been given in **[14]**.

We use this problem as the second and more complicated case of project scheduling for testing our self-adaptive GA. The solution representation and the alternatives of operators designed for the RIP/max are all applicable in this situation too. The only exception is that, in **[14]** a local search operator is extended for the RIPDCF/max which is applied on each solution with a specific probability. Therefore, the local search probability is another option for parameters which must be considered in the PAP of the algorithm.

a) OAP for RIPDCF/max

The only difference between OAP for the RIP/max and RIPDCF/max regards to the fitness calculation of solutions. Since the RIPDCF/max is to maximize the NPV of the project, the fitness of each solution is calculated as its NPV divided to the maximum NPV found during the whole run of GA.

E	**Source**	**#P**	**#A**	**#R**	**Average of RDP2** (%)				
					GA1	GA2	SS	SA	GA3
E1	Ref. **[14]**	60	10	1,3,5	0	0	5	0	0
		60	20	1,3,5	5	3	12	4	0
		60	30	1,3,5	11	3	16	4	1
E2	Ref. **[14]**	30	30	3,4,5	5	1	11	2	1
		30	60	3,4,5	7	4	17	5	1
E2	PROGEN	30	90	2,4	10	3	17	6	1
		30	120	2,4	12	4	19	5	1

TABLE 2. *Metaheuristic-based comparison results*

b) PAP for RIPDCF/max

Since there is a local search for the RIPDCF/max, the solution set of parameters has three cells for crossover, mutation, and local search probability. The cells are randomly generated from [0.6,1], [0,0.3], and [0,0.3], respectively. The other processes of PAP for RIPDCF/max are completely similar to the same process for RIP/max.

4. Comparison and Analysis

4.1 Comparison Results for RIP/max

In the project scheduling problem library *(PSPLIB)*, there are 810 RIP/max benchmarks solved by a branch and bound algorithm proposed in **[53]**. This branch and bound algorithm have found optimal solutions for 674 out of 810 test problems.

All of these 674 test problems are considered to be solved by our self-adaptive GA. **Table 1** shows the comparison results based on these benchmarks where RDP_1 is calculated according to *(8)*.

$$RDP_1 = \frac{\text{algorithm solution} - \text{exact solution}}{\text{exact solution}} \quad (8)$$

It is witnessed that, the algorithm is able to solve J10 group very well with an approximately zero deviation from the optimal solutions *(0.2% on average)*. It is also observed that, the proposed GA still performs well with a deviation below 2% for J20 and below 3% for J30, on average. In addition, the deviation seen for the most complex problems with 30 activities and 5 resources is less than 5%. Consequently, the results confirm the reliability of the proposed self-adaptive GA in tackling the RIP/max problems with a mean of RDP_1 less than 3% in all 674 benchmarks.

4.2 Comparison Results for RIPDCF/max

This section is classified to two parts. At first, the proposed GA is compared with four metaheuristics including two previous GAs, **[14]** and **[30]**, suggested for RIPDCF/max, a simulated annealing *(SA)*, and a scatter search *(SS)* algorithm designed in this research. Then, it is compared with near exact solutions found by the LINGO software.

No. of Problems	No. of Activities	No. of Resources	**Ave. of RDP_2 (%)**
60	10	1,3,5	0.09
39	20	1,3,5	0.83
16	30	1,3,5	1.32

TABLE 3. *Comparison to Lingo results*

The SA and SS are devised in the following way. The solution representation for both algorithms is similar to what used in the adaptive GA. In the SA, a neighborhood is produced by either a one-unit right shift in the starting time of a randomly selected activity or a one-unit left shift. In order to prevent the new child from being infeasible, after the right *(left)* shift, the starting time of its successors *(predecessors)* may be shifted forward or backward. In addition, the common cooling functions known as $T_i = kT_{i-1}$ is employed to decrease temperatures during the algorithm process where T_i shows the temperature of cooling process in period *i* and *k* is a coefficient. As for the SS, the initial population is generated similar to the adaptive GA. The distance matrix used in the SS is defined in *(9)* in which *d* denotes the distance value, h_1 and h_2 denote two population members, S_i shows the starting time of the activity *i, i=0, 1, ..., n*.

$$d_{P_1,P_2} = \frac{\sum_{i=0}^{n} \left| S_i^{h_1} - S_i^{h_2} \right|}{n} \quad (9)$$

For the solution combination, the uniform crossover designed for the adaptive GA is taken into account. When updating the reference sets, a new solution enters the reference set B_1, the set with best solutions of the population, if either the solution has both a better objective function than the worst objective function appeared in B_1 and a distance upon the threshold on the smallest distance to any member of B_1 or it has a better objective function only. In addition, a solution enters the reference set B_2, the set with diverse solutions of the population, if its distance is greater than the smallest distance in B_2.

a) Comparison to metaheuristics

The benchmarks employed here includes 180 RIPDCF/max instances available in **[14]**, 60 RIPDCF with simple finish to start precedence relations with zero time lags benchmarks available in **[14]**, and 60 RIPDCF benchmarks produced by PROGEN. Let the GA proposed in **[14]** be GA1, the GA introduced in **[30]** be GA2, and the present self-adaptive algorithm be GA3. **Table 2** shows the comparison results of these five algorithms experimented on 300 benchmarks. In **Table 2**, E shows the type of benchmark examples where E1 stands for RIPDCF/max and E2 for RIPDCF. In addition, #P, #A, and #R denotes the number of problems, activities, and resources, respectively. Comparisons are based on the relative deviation percentage shown in *(10)*, in which the best NPV for each benchmark is the best found among all of five algorithms.

$$RDP_2 = \frac{\text{best NPV} - \text{algorithm NPV}}{\text{algorithm NPV}} \quad (10)$$

Table 2 reveals that the self-adaptive GA is able to discover better solutions than all of other tested algorithms disregard of the type, size, or complexity of test problems. The deviation of the self-adaptive GA from the best points is approximately less than 1% in all levels of experimented problems while none of other algorithms were able find solutions with a deviation less than 1% except for the problems with 10 activities which is very simple and easy to solve. Looking at the average deviation of GA2 *(3%)* and GA3 *(1%)*, it is concluded that, the results of the present self-adaptive technique is even better than the well-known parameter tuning technique based on statistical analysis employed in GA2.

b) Comparison to near exact solutions

Although LINGO software may result in local optimal solutions for non-linear models like RIPDCF/max, its outcomes can be considered as near exact solutions useful for situations where no exact algorithm is available. According to **[14]**, LINGO was only able to solve 115 out of 180 RIPDCF/max test problems as presented in **Table 3**. We do a comparison based on these 115 test problems. Shortly, **Table 3** shows that the results of the proposed GA and LINGO differ only 0.75% on average.

5. Conclusion

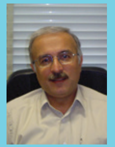
The GA is a powerful metaheuristic widely used to solve demanding PSPs. There are lots of alternatives for operators and parameters of GA which make GA practitioners confused how to choose the best ones. The self-adaptive parameter control is a common approach giving greater rewards to the better operators and parameters to participate more in the later processes of the GA. In this research, a new self-adaptive GA useful for PSPs was extended. The method has two parts coming after one another for several times during the search, providing input data for each other. One part is a self-adaptive mechanism to know the best operator combination. The other part is another self-adaptive mechanism to discover best values for parameters. The proposed GA was implemented on two class of PSPs namely the RIP/max and RIPDCF/max. The results of the proposed GA compared with exact solutions available in PSPLIB for the RIP/max showed a small deviation of 1.5%, on average. As for RIPDCF/max instances, the efficiency of method was measured by comparing with four other metaheuristics including two GAs available in the literature and two different metaheuristic algorithms *(SS and SA)* devised in this research. Comparisons showed that the proposed GA is far superior to others. In addition, the results of the algorithm were only 0.75% deviated from those of LINGO software in RIPDCF/max problems. Referring to all of these comparisons and evaluations, we conclude that the proposed GA is 1) more efficient than metaheuristic competitors, 2) less-deviated from optimal procedures, and 3) truly applicable on all type of PSPs.

For future research, the method can be applied on other combinatorial optimization problems.

authors

Amir Abbas Najafi received his B.S. degree in Industrial Engineering from Isfahan University of Technology in 1996, and his M.S. and Ph.D. degrees in Industrial Engineering from Sharif University of Technology in 1998 and 2005, respectively. He is currently an associate professor at K.N. Toosi University of Technology. His research interests include Applied Operations Research, Project Scheduling and Management and Portfolio Selection Models

Aria Shahsavar received his B.S. degree (2005) and M.S. degree (2008) both in Industrial Engineering from Islamic Azad University in Iran. He worked as an industrial engineer in car manufacturing industries from 2006 to 2009. Since then, he has been working in ship building industries as project planner and project manager. His research interests are Applied Operations Research, Project Scheduling and Management, and Metaheuristic algorithms.



Seyed Taghi Akhavan Niaki is Professor of Industrial Engineering at Sharif University of Technology. His research interests are in the areas of Quality Engineering, Simulation Modeling and Analysis, Applied Statistics, and Operations Research. Before joining Sharif University of Technology, he worked as a systems engineer and quality control manager for Iranian Electric Meters Company. He received his Bachelor of Science in Industrial Engineering from Sharif University of Technology in 1979, his Master's and his Ph.D. degrees both in Industrial Engineering from West Virginia University in 1989 and 1992, respectively. He is the Co-Chief-Editor-In-Chief of Scientia Iranica, the Editor of Scinetia Iranica Transactions E, the Acting Editor of Scientific and Research Journal of Sharif, a board member to several international journals, and a member of ϖμ.

references

[1] **L. Demeulemeester, and W. Herrolen, Project Scheduling a Research Handbook.** Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[2] **Kolisch, and S. Hartmann,** "Experimental investigation of heuristics for resource-constrained project scheduling: an update," Euro. J. Oper. Res. vol. 174, pp. 23-37, 2006.

[3] **Chan, D. K. H. Chua, and G. Kannan,** "Construction resource scheduling with genetic algorithms," J. Const. Eng. Manage. vol. 122, pp. 125–132, 1996.

[4] **Hartmann,** "A competitive genetic algorithm for resource-constrained project scheduling," Nav. Res. Log. vol. 45, pp. 733–750, 1998.

[5] **Hegazy,** "Optimization of resource allocation and leveling using genetic algorithms," J. Const. Eng. Manage. vol. 125, pp. 167-175, 1999.

[6] **Alcaraz, and C. Maroto,** "A robust genetic algorithm for resource allocation in project scheduling," Ann. Oper. Res. vol. 102, pp. 83–109, 2001.

[7] **Hartmann,** "A self-adapting genetic algorithm for project scheduling under resource constraints," Nav. Res. Log. vol. 49, pp. 433–448, 2002.

[8] **K. S. Hindi, H. Yang, and K. Fleszar,** "An evolutionary algorithm for resource-constrained project scheduling," IEEE T. Evolut. Comput. vol. 6, pp. 512–518, 2002.

[9] **Y. C. Toklu,** "Application of genetic algorithms to construction scheduling with or without resource constraints," Can. J. Civ. Eng. vol. 29, pp. 421–429, 2002.

[10] **A. Senouci, and N. Eldin,** "Use of genetic algorithms in resource scheduling of construction projects," J. Const. Eng. Manage. vol. 130, pp. 869-877, 2004.

[11] **A. A. Najafi, and S. T. A. Niaki,** "A genetic algorithm for resource investment problem with discounted cash flows," Appl. Math. Comput. vol. 183, pp. 1057–70, 2006.

[12] **S. Shadrokh, and F. Kianfar,** "A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty," Euro. J. Oper. Res. vol. 181, pp. 86–101, 2007.

[13] **J. Mendes, J. Goncalves, and M. Resende,** "A random key based genetic algorithm for the resource–constrained project scheduling problem," Comput. Oper. Res. vol. 36, pp. 92-109, 2009.

[14] **A. A. Najafi, S. T. A. Niaki, and M. Shahsavar,** "A parameter-tuned genetic algorithm for resource investment problem with discounted cash flows and generalized precedence relations," Comput. Oper. Res. vol. 36, pp. 2994-3001, 2009.

[15] **M. Shahsavar, S. T. A. Niaki, and A. A. Najafi,** "An efficient genetic algorithm to maximize net present value of project payments under inflation and bonus-penalty policy in resource investment problem," Adv. Eng. Softw. vol. 41, pp. 1023-1030, 2010.

[16] **B. Afshar-Nadjafi, and M. Arani,** "Multimode preemptive resource investment problem subject to due dates for activities: formulation and solution procedure," Advances in Operations Research. 2014, doi:10.1155/2014/740670.

[17] **M. Arjmand, and A. A. Najafi,** "Solving a multimode biobjective resource investment problem using metaheuristic algorithms," Advanced Computational Techniques in Electromagnetics. vol. 2015*(1)*, pp. 41-58.

[18] **J. Holland,** Adaptation in Natural and Artificial Systems. University of Michigan Press, Boston, 1975.

[19] **H. P. Schwefel,** Numerical Optimization of Computer Models. John Wiley & Sons, New York, 1981.

[20] **K. A. De Jong and W. M. Spears,** "An analysis of the interacting roles of population size and crossover in genetic algorithms," in Proceeding of the first Workshop on Parallel Problems Solving from Nature, H. P. Schwefel and R Manner Eds. Springer-Verlag, London, 1990, pp 38-47.

[21] **T. P. Bagchi, and K. Deb,** "Calibration of GA parameters: the design of experiments approach," Computer Science and Informatics. vol. 26, pp. 45-56, 1996.

[22] **A. W. M. Ng, and B. J. C. Perera,** "Selection of genetic algorithm operators for river water quality model calibration," Eng. Appl. Artif. Intell. vol. 16, pp. 529–541, 2003.

[23] **A. Czarn, C. MacNish, K. Vijayan, B. A. Turlach, and R. Gupta,** "Statistical exploratory analysis of genetic algorithms," IEEE T. Evolut. Comput. vol. 8, pp. 405–421, 2004.

[24] **C. B. Costa, M. M. R. Wolf, and F. R. Maciel,** "Factorial design technique applied to genetic algorithm parameters in a batch cooling crystallization optimization," Comput. Chem. Eng. vol. 29, pp. 2229–2241, 2005.

[25] **B. Adenso-Diaz, and M. Laguna,** "Fine-tuning of algorithms using fractional experimental designs and local Search," Oper. Res. vol. 54, pp. 99-114, 2006.

[26] **R. Ruiz, and C. Maroto,** "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," Euro. J. Oper. Res. vol. 169, pp. 781-800, 2006.

[27] **R. Ruiz, C. Maroto, and J. Alcaraz,** "Two new robust genetic algorithms for the flowshop scheduling problem," Omega. vol. 34, pp. 461-476, 2006.

[28] **C. B. Costa, E. A. Ccopa-Rivera, M. C. A. F. Rezende, M. M. R. Wolf, and F. R. Maciel** "Prior detection of genetic algorithm significant parameters: coupling factorial design technique to genetic algorithm," Chemical Engineering Science. vol. 62, pp. 4780-4801, 2007.

[29] **F. G. Lobo, C. F. Lima, and Z. Michalewicz,** Parameter Setting in Evolutionary Algorithms. Springer Velarg, Berlin, 2007.

[30] **M. Shahsavar, A. A. Najafi, and S. T. A. Niaki,** "Statistical design of genetic algorithms for combinatorial optimization problems," Math. Probl. Eng. vol. 2011*(2)*, pp. 1-17, doi:10.1155/2011/872415.

[31] **A. E. Eiben, R. Hinterding, and Z. Michalewicz** "Parameter control in evolutionary algorithms," IEEE T. Evolut. Comput. vol. 3, pp. 124–141, 1999.

[32] **T. Back,** "Optimal mutation rates in genetic search," in Proceedings of the Fifth International Conference on Genetic Algorithms, S. Forrest, Ed. San Mateo, CA, Morgan Kaufmann, 1993, pp. 2-8.

[33] **B. A. Julstrom,** "What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm," in Proceedings of the Sixth International Conference on Genetic Algorithms, L. Eshleman Ed. San Mateo, CA, Morgan Kaufmann, 1995, pp. 81-87.

[34] **W. M. Spears,** "Adapting crossover in evolutionary algorithms," in Evolutionary Programming IV. Proceedings of the Fourth Annual Conference on Evolutionary Programming, J. Mcdonnell, R. Reynolds, D. Fogel, Eds. MIT Press, Cambridge, MA, 1995, pp 367-84.

[35] **J. Lis,** "Parallel genetic algorithm with the dynamic control parameter," in Proceedings of IEEE International Conference on Evolutionary Computation. Nagoya, Japan, 1996, pp 324–329.

[36] **J. J. Grefenstette,** "Optimization of control parameters for genetic algorithms," IEEE Trans. Syst. Man. Cybern. vol. 16, pp. 122-128, 1986.

[37] **Q. T. Pham,** "Competitive evolution: a natural approach to operator selection," in Progress in Evolutionary Computation. Lecture Notes in Artificial Intelligence, X. Yao Ed. Springer-Verlag, Heidelberg, 1995, pp 49-60.

[38] **C. Chunlei and F. Yanjun,** "An adaptive mutation method for GA based on relative importance'" in Proceedings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering *(ICACTE)*. Chengdu, China, 2010, pp. 111-113.

[39] **G. Chen,** "Intelligent adaptive genetic algorithm and its application," in Proceedings of the 2011 International Conference on Intelligent Computation Technology and Automation *(ICICTA)*. 2011, pp. 163-166.

[40] **C. Jassadapakorn, and P. Chongstitvatana,** "Self-adaption mechanism to control the diversity of the population in genetic algorithms," International Journal of Computer Science & Information Technology. vol. 3*(4)*, pp. 111-127, 2011.

[41] **B. Tessema and G. G. Yen,** "A self adaptive penalty function based algorithm for constrained optimization," in Evolutionary Computation, 2006. CEC 2006. IEEE Congress on , Vancouver, BC , 2006, pp. 246 – 253.

[42] **R. Perzina, and J. Ramik,** "Timetabling problem with fuzzy constraints: a self-learning genetic algorithm," International Journal of Engineering and Innovative Technology. vol. 3*(4)*, pp. 105-113, 2013.

[43] **R. Perzina, and J. Ramik,** "Self-learning genetic algorithm for timetabling problem with fuzzy constraints," International Journal of Innovative Computing, Information and Control. vol. 9*(11)*, pp. 1349-4198, 2013.

[44] **A. F. Ali,** "Genetic local search algorithm with self-adaptive population resizing for solving global optimization problems. I.J. Information Engineering and Electronic Business. vol. 3, pp. 51-63, 2014.

[45] **F. Espinoza, B. S. Minsker, and D. Goldberg,** "An adaptive hybrid genetic algorithm for groundwater remediation design. J. Water Resour. Plann. Manage. vol. 131*(1)*, pp. 14–24, 2005.

[46] **F. Espinoza, and B. S. Minsker,** "Development of the enhanced self-adaptive hybrid genetic algorithm *(e-SAHGA)*," Water Resourc. Res. vol. 42*(8)*, 2006, doi:10.1029/2005WR004221.

[47] **T. E. Mihoub, L. Nolle, G. Schaefer, T. Nakashima, and A. Hopgood,** "A self-adaptive hybrid genetic algorithm for color clustering," in 2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, 2006, pp. 3158-3163.

[48] **T. Lili, K. Xiangdong, Z. Weimin, and Q. Feng,** "Modified self-adaptive immune genetic algorithm for optimization of combustion side reaction of p-Xylene oxidation. Chin. J. Chem. Eng. vol. 20*(6)*, pp. 1047-1052, 2012.

[49] **Y. Liu, Y. Feng, and P. Gilmore Pontius Jr,** "Spatially-explicit simulation of urban growth through self-adaptive genetic algorithm and cellular automata modeling," Land. vol. 3, pp. 719-738, 2014.

[50] **R. Zamani,** "A polarized adaptive schedule generation scheme for the resource-constrained project scheduling problem," RAIRO - Operations Research. vol. 46 *(1)*, pp. 23-39, 2012.

[51] **J. L. Ponz-Tienda, V. Yepes, E. Pellicer, and J. Moreno-Flores,** "The Resource Leveling Problem with multiple resources using an adaptive genetic algorithm," Autom. Constr. vol. 29, pp. 161–172, 2013.

[52] **R. H. Möhring,** "Minimizing costs of resource requirements in project networks subject to a fix completion time," Oper. Res. vol. 32, pp. 89-120, 1984.

[53] **K. Neumann, and J. Zimmermann,** "Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints," Euro. J. Oper. Res. vol. 127, pp. 425-443, 2000