Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2011, Article ID 872415, 17 pages
doi:10.1155/2011/872415

Research Article

Statistical Design of Genetic Algorithms for Combinatorial Optimization Problems

Moslem Shahsavar,[1] Amir Abbas Najafi,[2] and Seyed Taghi Akhavan Niaki[3]

[1] *Faculty of Industrial and Mechanical Engineering, Islamic Azad University, Qazvin Branch, Qazvin, P.O. Box 34185-1416, Iran*
[2] *Faculty of Industrial Engineering, K.N. Toosi University of Technology, Tehran 1999143344, Iran*
[3] *Department of Industrial Engineering, Sharif University of Technology, P.O. Box 11155-9414, Azadi Avenue, Tehran 1458889694, Iran*

Correspondence should be addressed to Moslem Shahsavar, m.shahsavar@ymail.com

Received 21 May 2011; Accepted 7 July 2011

Academic Editor: J. J. Judice

Copyright © 2011 Moslem Shahsavar et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Many genetic algorithms (GA) have been applied to solve different NP-complete combinatorial optimization problems so far. The striking point of using GA refers to selecting a combination of appropriate patterns in crossover, mutation, and and so forth and fine tuning of some parameters such as crossover probability, mutation probability, and and so forth. One way to design a robust GA is to select an optimal pattern and then to search for its parameter values using a tuning procedure. This paper addresses a methodology to both optimal pattern selection and the tuning phases by taking advantage of design of experiments and response surface methodology. To show the performances of the proposed procedure and demonstrate its applications, it is employed to design a robust GA to solve a project scheduling problem. Through the statistical comparison analyses between the performances of the proposed method and an existing GA, the effectiveness of the methodology is shown.

1. Introduction and Literature Survey

The combinatorial optimization involves problems in which their set of feasible solutions is discrete or can be reduced to a discrete one, and the goal is to find the best possible solution. In areas such as routing, task allocation, scheduling, and so forth, most of the problems are modelled in the form of combinatorial optimization problems.

Due to the NP completeness of many combinatorial optimization problems, they are quite difficult to be solved analytically, and exact search algorithms such as branch and bound may degenerate to complete enumeration, and the CPU time needed to solve them may grow exponentially in the worst case. To practically solve these problems, one has to be satisfied with finding good and approximately optimal solutions in reasonable, that is, polynomial time.

In recent decades, researchers have developed evolutionary algorithms to solve the combinatorial problems with practical sizes. Evolutionary algorithms (EA) form a class of search methods that work by incrementally improving the quality of a set of candidate solutions by variation and selection (Eiben and Smith [1]). Genetic algorithms suggested by Holland [2] is an evolutionary algorithm based on the principles of survival of the fittest and adaption originated of Darvin's evolution theorem. According to evolution theorem, new offspring is generated by joining old generations. An offspring with more fitness to environment is more able to survive and reproduce. The basic GA idea is as follow: the genetic pool of a given population potentially contains the solution, or a better solution, to a given adaptive problem. This solution is not "active" because the genetic combination on which it relies is split between several subjects. Only the association of different genomes can lead to the solution. This algorithm increases the fitness of solutions by the simulation of evolution process. For more details, see Goldberg [3] and Deb [4].

The two major steps in applying any GA to a particular problem are the specification of the representation and the evaluation (fitness) function (Deb [5]). These two items form the bridge between the original problem context and the problem-solving framework. The "chromosomes" encode a group of linked features of the problem, and the "Genes" encode the activation or deactivation of a feature. When defining a GA one needs to choose its components, such as variation operators (crossover, mutation, and recombination) that suit the representation, selection mechanisms for selecting parents and survivors, and an initial population (Ng and Perera [6]). Each of these components may have parameters, for instance, the probability of mutation, the problem of crossover, or the population size. The values of these parameters greatly determine whether the algorithm will find a near-optimum solution and whether it will find such a solution efficiently.

Evolution in GA is partly made at random, and the other part is based on both the behaviour of the applied patterns in component of GA and the setting desired values of the GA parameters. Therefore, the efficiency of a GA extremely relates to the selection of good patterns and tuning of its parameters. Many researchers have tried to optimize their GAs so far. In total, setting GA parameters are commonly divided into two cases: parameter control, and parameter tuning. In case of parameter control the parameter values are changing during a GA run. This requires initial parameter values and suitable control strategies, which in turn can be deterministic, adaptive, or self-adaptive (Eiben et al. [7]). For the case of parameter tuning, much attention has been focused on finding the theoretical relationship among these parameters. Schwefel [8] developed theoretical models for optimal mutation rates with respect to convergence and convergence rates in the context of function optimization.

De Jong and Spears [9] presented theoretical and empirical results on interacting roles of population size and crossover in genetic algorithm. As other approaches in optimization of GA parameters, Friesleben and Hartfelder [10] proposed a meta-GA approach in which both GA components and GA parameters are tuned. They demonstrated the importance of the right choice for the GA operators. Samples et al. [11] showed how parameter sweeps can be used for robustness and correlation analysis. Preuss and Bartz-Beielstein [12] embedded sequential parameter optimization in a wider framework of experimental EA design. However, majority of parameter tuning are based on design of experiment (DOE). Bagchi and Deb [13] proposed a DOE approach (a factorial design) to calibrate the parameters of

GA using pilot GA runs of relatively short length. As other implementations of DOE for parameter calibration, Ruiz and Maroto [14] and Ruiz et al. [15] used DOE to select the best GA component while previous works just considered DOE for setting parameter values. In addition, Saremi et al. [16] applied DOE to find the effects of individual parameters (factors) as well as to reveal any significant interaction among parameters. In the general field of experimental design, a paradigm shift that emphasizes a low cost of tuning over the performance of optimal parameter values was due to Taguchi and Wu [17]. For more studies on optimization of GA, parameters refer to François and Lavergne [18], Czarn et al. [19], Costa et al. [20, 21], and Lobo et al. [22].

While in the previous researches, there has been less attention in selecting the optimum or near the optimum pattern for the components of a GA and tuning its parameters simultaneously, in this paper, we attempt to introduce a new approach in which at first the effects of the GA components and parameters as well as their interactions are statistically analyzed. Then, in the second step, the optimal values of the parameters are discovered. Whereas the previous works have just assessed the effects of parameters while they were adjusted on some discrete points, in the present work we, search a continuous interval in order to find the optimal values of the parameters.

The organization of the rest of the paper is as follows. The description of the proposed methodology is given in Section 2. Section 3 contains the application of the proposed procedure in a project scheduling problem. The performance evaluation and statistical analyses of the proposed method applied to a project scheduling problem come in Section 4. Finally, Section 5 concludes and suggests some future areas of research.

2. The Proposed Methodology

A genetic algorithm is designed as a combination of patterns in encoding, generation, selection, joining, replacement, and stopping criteria. Although different patterns have been proposed to encode and to generate initial population, the relationships between the problem variables and the condition of the search range play an important role in designing the patterns. Nevertheless for other components of a GA, while a desirable combination is usually unknown, almost all existing patterns can be generally used. Furthermore, it seems that to design a robust GA, one needs to employ the following three phases:

Phase 1: designing different patterns to create the, algorithm

Phase 2: finding the significant combination of, and the designed patterns

Phase 3: tuning the parameters of the significant combination.

In the next subsections, we first review some of the most commonly used patterns of the GA components and then demonstrate the existing methods in the last two phases. At the end of this section, the proposed methodology will be described.

2.1. Phase 1: Designing Different Patterns to Create the Algorithm

In this section, we review some of the most commonly used patterns in GA components.

2.1.1. The Encoding Patterns

In a typical optimization problem, there exists a wide discrete/continuous range of points named search range, and the goal is to find a point that leads us to the optimal solution.

An essential characteristic of GAs is the encoding of the variables that describe the problem. There are many encoding patterns. Some patterns such as binary and gray encode the problem by a set of ones and zeros, and some others such as random keys, permutation, and real value encodings use numeric values to represent the chromosomes. For a review of encoding patterns, each with specific characteristic and capability, see (e.g. Jenkins [23], Hajela [24], Reeves [25]). Among these patterns, the real encoding has been shown to have more capability for complex problems (Andrzej [26]).

2.1.2. The Selection Patterns

Some of the most prevalent of the chromosomes' selection patterns proposed so far are based on the chromosomes' fitness and are the roulette selection (Michalewicz [27]) and the tournament selection (Back et al. [28]). Some other patterns do not deal with the chromosomes' fitness such as random selection in which the chromosomes are randomly selected and unlike selection in which all chromosomes attend the joining processes. A comparison of the selection varying schemes has been performed by Goldberg and Deb [29].

2.1.3. The Crossover Patterns

One of the most important components of a GA is the crossover operator. There are two broad classes of the crossover operation, namely, the point crossovers (Deb [5]) and the uniform crossover (Syswerda [30]). In a point crossover, the string of the parents' chromosomes is randomly cut on one or more points. Then by replacing the existing genes between the cut points of the parents' chromosomes, a new child is created. However, in a uniform crossover, any genes of both parents' chromosomes have a chance to attend the child's chromosome.

Traditionally, GAs have relied upon point crossovers. However, there are many instances in which having a higher number of crossover points is beneficial (Eshelman et al. [31]). Furthermore, perhaps the most surprising result is the effectiveness of the uniform crossover. Uniform crossover produces, on average, $L/2$ crossings on string length L (Spears and De Jong [32]). In addition to empirical investigations, considerable effort has been directed towards the theoretical comparisons between different crossover operators (De Jong and Spears [33]). Nonetheless, these theories are not sufficiently general to predict when to use crossover, or what form of crossover to use.

2.1.4. The Mutation Patterns

Another important operator of a GA is mutation that acts like an insurance policy against premature loss of important information (Goldberg [3]). Since the mutation is a structural change in a set of candidate genes of a chromosome, it is directly related to the designed encoding pattern. For example, in a binary encoding, mutation can be defined as changing the value of a gene from one to zero. However, two common viewpoints exist to perform the mutations that are not related to the encoding pattern. In the first one, a mutation probability is assigned to each child. Whenever the mutation is going to be performed, it just has to decide to mutate all of the candidate genes of a child or none. In the second one, a mutation probability is assigned to each candidate gene. Then, through each mutation operation, the algorithm faces a number of decision-making steps that is equal to the number of the candidate genes. In fact, for each candidate, the algorithm has to assess whether the mutation is performed or not. Therefore, it usually occurs that some of the candidates have been mutated and the rest stay without any change.

2.1.5. The Replacement Patterns

In a GA, the children of the old generation are replaced with the newly generated children. A variety of methods have been proposed in the literature for this task. In one of these methods called (M, L), in each generation the M existing chromosomes generate L new children and then through L-generated children M new children are chosen for the next generation. It is obvious that in this method L must be greater than M. Otherwise, after some generations the population size may become zero. In the second method named $(M + L)$, the M existing chromosomes in each generation create L new children and then through these $M + L$ available chromosome, M children are chosen. Michalewicz [27] experimentally showed that the (M, L) method performs better than the $(M + L)$ method in terms of the solution quality. In another replacement method named Elitism, few of the most fitted children are directly reproduced to the next generation, and the other members of the new generation are created based on the genetic operators. This replacement method has the potential of converging to the global optimal effectively. When this method is used, the number/percentage of the reproduced children can be considered as one of the GA parameters. Finally, another replacement strategy is called preselection in which if any child shows better fitness than its parents, it is replaced with the one with better fitness (Goldberg [3]).

2.1.6. The Stopping Patterns

The stopping criteria are usually of two types. The first one (such as either reaching a predefined number of iteration or a predefined iteration time of running GA) is called the passive stopping criterion in which the algorithm independently of the obtained results is stopped. The second one (such as either reaching a unique solution in some sequential generations or the difference between the average fitness of some sequential generations is low) is called the sequential criterion in which stopping the GA depends on the quality of the obtained results. Both of these criteria may be treated as the GA parameters.

In summary, some of the predetermined patterns along with their parameters and symbols used to describe them are given in Table 1. It should be noted that selecting the significant combination of the patterns among more designed patterns would lead to more accurate basis of designing a robust GA. Many of these patterns have the same structure such that they can be transformed to each other during their designing phase. In this case, the pattern creation and designing do not need too much times and efforts.

2.2. Phase 2: Finding the Significant Combination of the Designed Patterns

While the performances of a GA (responses) are affected by some controllable factors such as encoding, selection, crossover, mutation, replacement, and stopping patterns, as well as uncontrollable factors in which their different combinations would result in different performances, design of experiments can be implemented to study the effects of the input factors and their interactions on the system performances and delineate which factor(s) has (have) the most effect(s) on the response(s). Furthermore, when the study involves two or more factors, factorial designs are generally the most efficient way of delineating the significant factors (Montgomery [34]).

Table 1: Common patterns in designing GA components and their parameters.

Component	Operators	Parameters	Symbols
Parent selection	Roulette	—	A_1
	Tournament	Tour size	A_2
	Random	—	A_3
	Unlike	—	A_4
Recombination	One point	Crossover probability	B_1
	Two point	Crossover probability	B_2
	Uniform	Crossover probability	B_3
Mutation	Prob. to any child	Mutation probability	C_1
	Prob. to any gene	Mutation probability	C_2
Replacement methods	$(M + L)$	L	D_1
	(M, L)	L	D_2
	Elitism	Number of chromosomes for reproduction	D_3
	Preselection	—	D_4
Stopping criteria	Passive	Number of iteration/time	E_1
	Sequential	Number of the successive iterations with the same best solution	E_2

In designing a robust GA, the effects of different patterns and parameters along with their interactions can be analyzed using a factorial design such as a full factorial design (a 2^k factorial), or a 2^{k-p} fractional factorial design, in which k is the number of factors and p represents a fraction. If the results of the analysis of variance table of such designs indicate statistical significance, then a post hoc analysis may be performed to assess the important factor(s). The least significant difference (LSD) and the Duncan tests are among the tests that compare all paired factor means and delineate the important factor(s) (Montgomery [34]). Either of these tests as well as others may be employed to find the important patterns affecting the efficiency of the designed GA.

2.3. Phase 3: Tuning the Parameters of the Significant Pattern Combination

Response surface methodology (RSM) is a collection of statistical and mathematical techniques useful for modelling and analysis of problems in which responses affected by several variables, and its goal is to optimize these responses. In most of these problems, the relation between the response and the factors (independent variables) is not definite, requiring to first estimate this relationship. To do this, if there exists no curvature in the response surface, one can employ the first order model; otherwise, a model of higher order (usually a second order model) should be used. Then, a level combination of the factors should be found such that the optimal value of the response is reached. To do this, the deepest ascent method (for maximization problem) or the deepest descent method (for minimization problem) are employed to move on a path with the most gradient that makes the most increase or decrease in the objective function. A comprehensive survey of RSM refers to Myers and Montgomery [35].

In order to tune the parameters of the designed GA, one may use RSM. If the relation between the algorithm efficiency and the important GA parameters is linear, then depending on the constraints (time and cost) on the size of experiments either the 2^k factorial or 2^{k-p}

fractional factorial design is used. In case the relation is nonlinear, the central composite design (CCD) or other desirable designs can be applied to estimate the response function. The response surfaces for designing a robust GA may be defined as (a) minimization of the relative deviation percentage (RDP) obtained by employing the algorithm to the optimal result and (b) minimization of the algorithm runtimes or optimization of another goal set by the algorithm designers. In case of multiple goals, the multiobjective optimization methods such as goal programming (GP), fuzzy goal programming (FGP) (Narasimhan [36], Tiwari et al. [37]), or the other known methods can be applied.

The present methodology can be carried out independent of the selected encoding pattern. This is due to the fact that the patterns work independent of encoding as they use the fitness of chromosomes. In other words, regardless of the encoding patterns, we need to have a method to calculate the fitness of chromosomes. There is also a similar consequence for the replacement patterns. They all are either completely free of encoding type or in some situations such as Elitism they are only associated with the fitness. For the most important part of GA, that is, the crossover pattern, since some characteristics (genes) of patterns are replaced through the point and the uniform crossovers, the methodology is applicable in all encoding patterns. The only part of GA that is associated with the encoding system is mutation. As explained in Section 2.1.4, there are two strategies regulating how the designed mutation is applied. In this case, since different patterns can be defined through the encoding type, the present methodology can be still applicable.

In the next section, a project scheduling problem is used to demonstrate the steps involved in the proposed methodology along with its possible implementation in practice.

3. Application of the Proposed Methodology in a Project Scheduling Problem

Project scheduling problem is an important branch of combinatorial optimization problem in which the goal is optimization of one or some objectives subject to constraints on the activities and the resources. Due to its NP-Hardness structure, researchers have widely considered metaheuristics, particularly genetic algorithms, in order to solve it. Among these research works, different chromosome encodings and parameter settings have been introduced so far. For example, the priority-value based and priority rule-based representations were developed by Lee and Kim [38] and Özdamar [39], respectively. Hartmann [40] and Alcaraz and Maroto [41] used permutation-based encoding. Furthermore, Alcaraz and Maroto [41] compared different selection operators and showed the best is the 2-tournament operator in which two individual solutions are selected randomly from the current population, and the best fitted is selected as a parent-solution. Gonçalves et al. [42] presented a genetic algorithm for the resource constrained multiproject scheduling problem based on random keys for the chromosome encoding. To adjust the parameters, they investigated four factors: (1) the top percentage from the previous population chromosomes that are copied to the next generation (TOP), (2) the bottom percentage of the population chromosomes that are replaced with randomly generated chromosome (BOT), (3) the crossover probability (CProb), and (4) the population size. A small pilot study was performed by experimenting the combinations of these factors.

Resource investment problem with discounted cash flows introduced by Najafi and Niaki [43] is a branch of project scheduling problems in which the projects are mathematically formulated so that the resource levels and starting times of activities are the decision

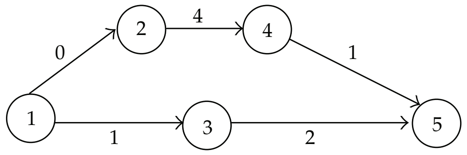


Figure 1: Graph of the example network.

variables. Since the cash flows occur during the project, its goal is maximization of net present value of the project cash flows. This is an NP-hard model and cannot be easily solved. Najafi et al. [44] developed a model of this problem by minimal and maximal time lags and suggested a genetic algorithm to solve it. In their GA, a real-valued encoding has been used to represent the chromosomes. In this regard, each individual chromosome, $I = (S_1, S_2, \ldots, S_n)$, is a vector consisting n genes where n denotes the number of the project activities. Each position in the chromosome denotes the corresponding activity number, and the value of gene i states the starting time of activity i at the precedence feasible schedule of the chromosome. For example, consider a project whose network is depicted in Figure 1. The values located in the nodes indicates the activity numbers and the ones associated with the arcs connecting activity i to activity j display the minimum time lag between start to start activity i as predecessor of activity j. For the network shown in Figure 1, $I = (0, 0, 2, 5, 6)$ is a chromosome showing a real schedule for starting time of the activities.

Since Andrzej [26] showed that the real encoding has more capability for complex problems and the algorithm of Najafi et al. [44] has a real-value based representation for chromosomes, in this research, we decided to implement them as well. Shortly, the tournament selection pattern (A_2) was the choice of selecting the parents. Moreover, the crossover operation has been performed in a combination method involving two operators (one-point crossover (B_1) and uniform crossover (B_2)). For mutation operation, assigning a probability to any chromosome (C_1) and for children replacement, the preselection method (D_4) has been selected. In addition, to stop the algorithm a combination of two criteria has been applied, a predefined number of iteration along with a number of sequence generations with the same good solution. All of the patterns used in their algorithm have been reached based on a trial and error method. They tested the algorithm on 180 test problems included 10, 20, and 30 nondummy activities with 1, 3, and 5 resources and compared their GA solutions with the ones obtained by solving the mathematical models of the problem.

3.1. Phase 1: Designing Different Patterns

Since having more patterns are desired for a robust GA, in addition to the ones devised by Najafi et al. [44], we design more pattern combinations in the algorithm efficiency comparison study. For this purpose, one other selection pattern, the unlike selection (A_4), assigning a probability to any candidate gene for mutation (C_2), and the Elitism replacement method (D_3) are also employed. The tournament and the unlike patterns are the two levels of factor A. Furthermore, both one-point and uniform crossover operators are considered in the study. While the one-point and the uniform crossover operators were considered as a combined factor in Najafi et al. [44] method, in this research they are the two levels of factor B. The probability assignment to both the children and the genes are the two levels of factor C. The two levels of factor D are the preselection and the Elitism. Factor E has two levels of the number of iterations (passive stopping criterion) and reaching a good solution in three

Table 2: The coded factor levels.

Factors	Coded levels	
	High level (+1)	Low level (−1)
A	Tournament	Unlike
B	One-point crossover	Uniform crossover
C	A probability to any child	A probability to any gene
D	Preselection	Elitism
E	Passive	Sequential
F	0.9	0.7
G	0.1	0.05
H	0.1	0.05

consecutive generations (sequential criterion). Finally, factors F, G, and H, each with two quantitative levels, denote the crossover probability, the mutation probability, and the local improvement probability, respectively. The levels of factor F, G, and H are $\{F_1 = 0.7, F_2 = 0.9\}$, $\{G_1 = 0.05, G_2 = 0.1\}$, and $\{H_1 = 0.05, H_2 = 0.1\}$, respectively. Table 2 shows the factor levels under study.

3.2. Phase 2: Finding the Significant Combination

One run of the full factorial design of the 8 aforementioned factors requires $2^8 = 256$ experiments. Due to the time and cost, constraints suppose a design with smaller run numbers is desired. Whereas all of these factors contain 2 levels, a 2^{k-p} fractional factorial design is a good option. Even if factors with more than two levels exist in the experiment, either a higher-order fractional factorial or a general factorial design may be used. In this study, a 2^{8-4}_{IV} fractional factorial design containing 16 runs with generators $E = BCD$, $F = ACD$, $G = ABC$, and $H = ABD$, is used to test the performances of the proposed methodology on the 30 test problems with known optimal solution.

For the analysis of variance, the relative deviation percentage (RDP) of the proposed GA solutions to the optimal solutions which is calculated according to (3.1) was used as the response.

$$\text{RDP} = \frac{\text{Proposed GA Solution} - \text{Optimal Solution}}{\text{Optimal Solution}}. \tag{3.1}$$

Although the probability distribution of the response is not known, since 30 test problems is used in each of which the summation of the responses are used for different factor combinations, the central limit theorem implies an approximate normal distribution. Assuming that more than 2-way interaction effects are not significant, the analysis of variance of this experiment is given in Table 3. The results of Table 3 show that the main effects, along with the 2-way interaction effects, are statistically significant.

Table 3: Analysis of variance for RDP.

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main effects	8	9.464	9.464	1.18294	17.74	0
2-way interactions	7	2.804	2.804	0.40054	6.01	0
Residual error	464	30.933	30.933	0.06667		
Pure error	464	30.933	30.933	0.06667		
Total	479	43.2				

Table 4: The search ranges for input variables.

Parameters	Ranges
Population size	n–$2n$
Crossover probability	0.8–1
Mutation probability	0.025–0.075
Local improvement probability	0.05–0.15

In order to find the specific significant effects, the Duncan's multiple range test as one of the ad hoc analysis methods results in the following ranking:

$$\begin{gathered} A^{-1} \succ A^{+1}; \quad B^{-1} \succ\succ B^{+1}; \quad C^{-1} \succ C^{+1}; \quad D^{-1} \succ D^{+1} \\ E^{+1} \succ\succ E^{-1}; \quad F^{+1} \succ F^{-1}; \quad G^{-1} \succ G^{+1}; \quad H^{+1} \succ H^{-1}. \end{gathered} \tag{3.2}$$

In which "≻" shows the better and "≻≻" shows the statistically better, and "−1" and "+1" of the superscripts denote the low and the high levels of the corresponding factor. Thus, the factor combination $A^{-1}B^{-1}C^{-1}D^{-1}E^{+1}$ is known as the significant combination and $F^{+1} = 0.9$, $G^{-1} = 0.05$, and $H^{+1} = 0.1$ are known as the neighbour points of the optimal values of some important parameters of the GA that needs to be considered to be tuned in the next phase.

3.3. Phase 3: Tuning the Parameters

The candidate parameters in the tuning process are population size (X_1), crossover probability (X_2), mutation probability (X_3), and local improvement probability (X_4). Table 4 shows the search ranges of these parameters. The lower and the upper points of the ranges are considered as the lower and the upper levels of these factors and coded as ±1. Furthermore, n shows the number activities of the project.

In the tuning phase, a 2^4 central composite factorial (CCF) design with 4 central points and 8 axial points of $(\pm1, 0, 0, 0,)$, $(0, \pm1, 0, 0)$, $(0, 0, \pm1, 0)$, and $(0, 0, 0, \pm1)$ including 28 combinations of the factor levels is employed. Table 5 shows the factor level combinations along with their corresponding responses.

The first response (Y_1) denotes the mean RDP, and the second response (Y_2) represents the mean ratio of the proposed GA CPU-times to the required optimal CPU-times, where the optimal CPU-time of each problem is defined as the minimum time obtained in 28 executed levels.

The results of Table 5 are used to estimate both functions $\widehat{Y}_1$ (accuracy of the solution) and $\widehat{Y}_2$ (quality of the solution) using SAS software. These estimates are given in (3.3) and (3.4), respectively,

$$\begin{aligned}\widehat{Y}_1 = & -0.016 + 0.004X_1 + 0.001X_2 + 0.0001X_3 + 0.003X_4 - 0.001X_1^2 + 0.0004X_2^2 \\ & - 0.002X_3^2 - 0.0004X_4^2 - 0.0009X_1X_2 - 0.0006X_1X_3 - 0.002X_1X_4 \\ & - 0.0005X_2X_3 + 0.001X_2X_4 + 0.0003X_3X_4,\end{aligned} \tag{3.3}$$

$$\begin{aligned}\widehat{Y}_2 = & \ 0.4 - 0.23X_1 - 0.06X_2 - 0.007X_3 - 0.07X_4 + 0.11X_1^2 + 0.009X_2^2 - 0.012X_3^2 \\ & + 0.013X_4^2 + 0.026X_1X_2 + 0.002X_1X_3 + 0.03X_1X_4 - 0.001X_2X_3 \\ & + 0.005X_2X_4 - 0.003X_3X_4.\end{aligned} \tag{3.4}$$

Tables 6 and 7 contain the analysis of variance results that show significant linear, quadratic, and interaction effects for both the solution accuracy and quality at 90% confidence level.

Since the goal is to find the parameters values such that both objective functions are simultaneously optimized, a bi-objective optimization problem is needed to be solved. The fuzzy goal-programming (FGP) technique transforms the multiobjective decision-making problem to a single objective using fuzzy set theory. In FGP, a membership function is defined for each objective (Y_j) based on (3.5), in which U_j is its objective value when it is optimized alone, and L_j is its worst objective value considering the values of the variable set obtained in other objectives when they are optimized separately. The goal is to reach $\mu_{Y_j} = 1$. However, since conflict may happen by the other goals, FGP tries to maximize the achievement degree of μ_{Y_j} to resolve the conflict

$$\mu_{Y_j} = \begin{cases} 0 & Y_j < L_j, \\ \dfrac{Y_j - U_j}{U_j - L_j} & L_j \le Y_j \le U_j, \\ 1 & Y_j > U_j. \end{cases} \tag{3.5}$$

Suppose α_j to be the achievement degree of μ_{Y_j}. The single objective function in FGP is then the maximization of the weighted sum of the achievement degrees ($\sum_{j=1}^{k} w_j \alpha_j$) of the goals ($\mu_{Y_j}s$) subject to the existing constraints and the ones added for the membership functions (e.g., $\mu_{Y_j} \ge \alpha_j$ for objective Y_j).

To solve this problem, we first obtain the pay-off table of the positive ideal solution (PIS) as shown in Table 8.

Table 5: The results of the CCF design.

	Input variables				Responses	
Runs	X_1	X_2	X_3	X_4	Y_1	Y_2
(1)	0	0	1	0	−0.02	0.38
(2)	1	0	0	0	−0.02	0.29
(3)	0	−1	0	0	−0.02	0.46
(4)	−1	0	0	0	−0.02	0.75
(5)	1	1	−1	1	−0.01	0.23
(6)	1	−1	1	−1	−0.02	0.37
(7)	−1	1	1	1	−0.02	0.57
(8)	1	−1	1	1	−0.01	0.28
(9)	0	0	0	−1	−0.02	0.48
(10)	0	1	0	0	−0.01	0.37
(11)	−1	−1	1	1	−0.02	0.73
(12)	0	0	0	0	−0.02	0.41
(13)	0	0	0	0	−0.01	0.41
(14)	1	1	1	1	−0.01	0.23
(15)	0	0	0	1	−0.01	0.36
(16)	1	1	1	−1	−0.02	0.3
(17)	−1	−1	−1	−1	−0.03	0.96
(18)	1	1	−1	−1	−0.01	0.31
(19)	0	0	0	0	−0.01	0.41
(20)	−1	1	−1	1	−0.01	0.61
(21)	−1	−1	−1	1	−0.02	0.76
(22)	−1	−1	1	−1	−0.03	0.96
(23)	1	−1	−1	−1	−0.01	0.38
(24)	−1	1	−1	−1	−0.03	0.78
(25)	1	−1	−1	1	−0.01	0.29
(26)	0	0	−1	0	−0.02	0.41
(27)	0	0	0	0	−0.02	0.42
(28)	−1	1	1	−1	−0.03	0.78

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

$$\mu_{Y_1} = \begin{cases} 0 & Y_1 < -0.029, \\ \dfrac{Y_1 + 0.009}{-0.009 + 0.029} & -0.029 \le Y_1 \le -0.009, \\ 1 & Y_1 > -0.009, \end{cases}$$
$$\mu_{Y_2} = \begin{cases} 0 & Y_2 < 0.24, \\ \dfrac{Y_2 - 0.97}{0.97 + 0.24} & 0.24 \le Y_2 \le 0.97, \\ 1 & Y_2 > 0.97. \end{cases} \tag{3.6}$$

Table 6: Analysis of variance for solution accuracy.

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Regression	14	0.000712	0.000712	0.000051	9.28	0
Linear effect	4	0.000551	0.000551	0.000138	25.11	0
Quadratic effect	4	0.000056	0.000056	0.000014	2.54	0.09
Interaction	6	0.000106	0.000106	0.000018	3.21	0.037
Residual error	13	0.000071	0.000071	0.000005		
Lack of fit	10	0.000058	0.000058	0.000006	1.32	0.458
Pure error	3	0.000013	0.000013	0.000004		
Total	27	0.000783				
$S = 0.002342$		R-Sq = 90.9%			R-Sq (adj) = 81.1%	

Table 7: Analysis of variance for solution quality.

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Regression	14	1.25963	1.25963	0.089973	720.15	0
Linear effect	4	1.1406	1.1406	0.285149	2282.35	0
Quadratic effect	4	0.09321	0.09321	0.023302	186.51	0
Interaction	6	0.02582	0.02582	0.004304	34.45	0
Residual error	13	0.00162	0.00162	0.000125		
Lack of fit	10	0.00156	0.00156	0.000156	7.27	0.065
Pure error	3	0.00006	0.00006	0.000021		
Total	27	1.26125				
$S = 0.01118$		R-Sq = 99.9%			R-Sq (adj) = 99.7%	

Table 8: Payoff table of PIS.

	Y_1	Y_2
Max Y_1	−0.009	0.24
Max Y_2	−0.029	0.97

Then, the model becomes

$$\begin{aligned}
&\max \quad Z = \sum_{j=1}^{2} w_j \alpha_j \\
&\text{s.t.} \quad \mu_{Y_j} \geq \alpha_j; \quad j = 1, 2; \\
&-1 \leq X_i \leq 1; \quad i = 1, \ldots, 5; \\
&\alpha_j \in [0, 1]; \quad j = 1, 2.
\end{aligned} \tag{3.7}$$

The optimal values obtained by the FGP method with $w_1 = 0.75$ and $w_2 = 0.25$ for Y_1 and Y_2, respectively, are shown in Table 9.

Table 9: The optimal values of the parameters.

Parameter	Optimum value
Population size	$1.56n$
Crossover probability	1
Mutation probability	0.048
Local improvement probability	0.15

Table 10: Comparison results.

No. of activities	No. of problems	A	B	C	D
10	60	0.2%	3	2	−22%
20	60	2.4%	37	30	−13%
30	60	8.2%	102	94	−7%

4. Performance Analyses

To compare the performance of the proposed optimized genetic algorithm, all 180 problems used in Najafi et al. [44] are applied by the new algorithm as well. Then, statistical tests are used to compare the results. The results of the comparison experiments on the test problems are shown in Table 10.

The following notations are used in Table 10.

A: Average relative deviation percentages of the optimized GA solution to the GA of Najafi et al. [44] solution.

B: Average runtime (in seconds) required to obtain the solutions by GA of Najafi et al. [44].

C: Average runtime (in seconds) required to obtain the solutions by the optimized GA.

D: Average relative-deviation-percentages of the optimized GA runtime to the GA of Najafi et al. [44] runtime.

These results of Table 10 show that the optimized GA performs better than the GA of Najafi et al. [44] solutions (on the average 3.6% better) with less amount of required CPU time (on the average −14%). Furthermore, two tests of hypothesis are performed to prove the statistically better performances of the proposed methodology. The results of the tests that are given in Tables 11 and 12 show that the performances of the proposed optimized GA is better than those of the GA of Najafi et al. [44].

5. Conclusion

The genetic algorithm is known as one of the most robust and efficacious methods to solve combinatorial optimization problems and has been widely used in recent researches. Since different viewpoints suggested to design this algorithm and its parameters greatly affect the solution quality, in this research a methodology that contains three phases has been proposed to design an optimal genetic algorithm. The phases are designing different combinations of common viewpoints to create GA, selection of the significant combinations by design of experiments, and tuning the parameters using response surface methodology. This methodology was then applied to optimize a GA to solve a project scheduling problem. Statistical

Table 11: Hypotheses test of $\mu = 0$ versus $\mu > 0$ for RDP obtained from solutions of the optimized GA to the GA of Najafi et al. [44].

Variable	N	Mean	StDev	SE mean	95% lower bound	T	P
RDP Solution	180	0.035769	0.164963	0.012296	0.015439	2.91	0.002

Table 12: Hypotheses test of $\mu = 0$ versus $\mu < 0$ for the RDP obtained from runtimes of the optimized GA to the GA of Najafi et al. [44].

Variable	N	Mean	StDev	SE mean	95% lower bound	T	P
RDP Runtime	180	−0.1425	0.395694	0.029493	−0.09374	−4.83	0

comparisons of the results obtained by the proposed GA with the ones from an existing GA to solve the project scheduling problem verified better performances of the new algorithm with less amount of required CPU time. The proposed methodology can be employed for other encoding patterns of GA in addition to other metaheuristic algorithms as future researches.

References

[1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, Germany, 2003.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Boston, Mass, USA, 1975.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, Mass, USA, 1989.

[4] K. Deb, "An introduction to genetic algorithms," *Sādhanā. Academy Proceedings in Engineering Sciences*, vol. 24, no. 4-5, pp. 293–315, 1999.

[5] K. Deb, "Genetic algorithm in search and optimization: the technique and applications," in *Proceedings of the International Workshop on Soft Computing and Intelligent Systems*, pp. 58–87, Calcutta, India, 1998.

[6] A. W. M. Ng and B. J. C. Perera, "Selection of genetic algorithm operators for river water quality model calibration," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 5-6, pp. 529–541, 2003.

[7] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

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

[9] K. A. de Jong and W. M. Spears, "An analysis of the interacting roles of population size and crossover in genetic algorithms," in *Proceedings of the International Conference on Parallel Problems Solving from Nature*, pp. 38–47, Springer, Berlin, Germany, 1990.

[10] B. Friesleben and M. Hartfelder, "Optimization of genetic algorithms by genetic algorithms," in *Artificial Neural Networks and Genetic Algorithms*, R. F. Albrecht, C. R. Reeves, and N. C. Steele, Eds., pp. 392–399, Springer, Belin, Germany, 1993.

[11] M. E. Samples, M. J. Byom, and J. M. Daida, "Parameter sweeps for exploring parameter spaces of genetic and evolutionary algorithms," in *Parameter Setting in Evolutionary Algorithms*, F. G. Lobo, C.F. Lima, and Z. Michalewicz, Eds., pp. 161–184, Springer, Berlin, Germany, 2007.

[12] M. Preuss and T. Bartz-Beielstein, "Sequential parameter optimization applied to self-adaptation for binary-coded evolutionary algorithms," in *Parameter Setting in Evolutionary Algorithms*, F. G. Lobo, C.F. Lima, and Z. Michalewicz, Eds., pp. 91–119, Springer, Berlin, Germany, 2007.

[13] T. P. Bagchi and K. Deb, "Calibration of GA parameters: the design of experiments approach," *Computer Science and Informatics*, vol. 26, no. 3, pp. 45–56, 1996.

[14] R. Ruiz and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *European Journal of Operational Research*, vol. 169, no. 3, pp. 781–800, 2006.

[15] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega*, vol. 34, no. 5, pp. 461–476, 2006.

[16] A. Saremi, T. Y. ElMekkawy, and G. G. Wang, "Tuning the parameters of a memetic algorithm to solve vehicle routing problem with backhauls using design of experiments," *International Journal of Operations Research*, vol. 4, pp. 206–219, 2007.

[17] G. Taguchi and Y. Wu, *Introduction to Off-Line Quality Control*, Central Japan Quality Control Association, Nagoya, Japan, 1980.

[18] O. François and C. Lavergne, "Design of evolutionary algorithms—a statistical perspective," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 2, pp. 129–148, 2001.

[19] A. Czarn, C. MacNish, K. Vijayan, B. Turlach, and R. Gupta, "Statistical exploratory analysis of genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 4, pp. 405–421, 2004.

[20] C. B. B. Costa, M. R. W. Maciel, and R. M. Filho, "Factorial design technique applied to genetic algorithm parameters in a batch cooling crystallization optimisation," *Computers and Chemical Engineering*, vol. 29, no. 10, pp. 2229–2241, 2005.

[21] C. B. B. Costa, E. A. Ccopa-Rivera, M. C. A. F. Rezende, M. R. W. Maciel, and R. M. Filho, "Prior detection of genetic algorithm significant parameters: coupling factorial design technique to genetic algorithm," *Chemical Engineering Science*, vol. 62, no. 17, pp. 4780–4801, 2007.

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

[23] W. M. Jenkins, "Towards structural optimization via the genetic algorithm," *Computers and Structures*, vol. 40, no. 5, pp. 1321–1327, 1991.

[24] P. Hajela, "Stochastic search in discrete structural optimization simulated annealing, genetic algorithms and neural networks," in *Discrete Structural Optimization*, W. Gutkowski, Ed., pp. 55–134, Springer, New York, NY, USA, 1997.

[25] G. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley & Sons, New York, NY, USA, 1993.

[26] O. Andrzej, *Evolutionary Algorithms for Single and Multicriteria Design Optimization*, Physica, New York, NY, USA, 2002.

[27] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, London, UK, 1996.

[28] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, Institute of Physics Publishing, Bristol, UK, 1997.

[29] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*, G. Rawlins, Ed., pp. 69–93, Morgan Kaufmann, San Mateo, Calif, USA, 1991.

[30] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd Conference on Genetic Algorithms*, J. D. Schaffer, Ed., pp. 2–8, San Francisco, Calif, USA, 1989.

[31] L. Eshelman, R. Caruana, and D. Schaffer, "Biases in the crossover landscape," in *Proceedings of the 3rd Conference on Genetic Algorithms*, J. D. Schaffer, Ed., pp. 10–19, San Francisco, Calif, USA, 1989.

[32] W. M. Spears and K. A. de Jong, "On the virtues of parameterized uniform crossover," in *Proceedings of the 4rd Conference on Genetic Algorithms*, J. D. Schaffer, Ed., pp. 230–236, San Francisco, Calif, USA, 1991.

[33] K. A. de Jong and W. M. Spears, "A formal analysis of the role of multi-point crossover in genetic algorithms," *Annals of Mathematics and Artificial Intelligence*, vol. 5, no. 1, pp. 1–26, 1992.

[34] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, New York, NY, USA, 6th edition, 2005.

[35] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley & Sons, New York, NY, USA, 2002.

[36] R. Narasimhan, "Goal programming in a fuzzy environment," *Decision Sciences*, vol. 11, pp. 325–336, 1980.

[37] R. N. Tiwari, S. Dharmar, and J. R. Rao, "Fuzzy goal programming—an additive model," *Fuzzy Sets and Systems*, vol. 24, no. 1, pp. 27–34, 1987.

[38] J. K. Lee and Y. D. Kim, "Search heuristics for resource constrained project scheduling," *Journal of the Operational Research Society*, vol. 47, no. 5, pp. 678–689, 1996.

[39] L. Özdamar, "A genetic algorithm approach to a general category project scheduling problem," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 29, no. 1, pp. 44–59, 1999.

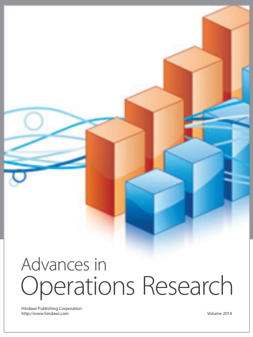
[40] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Naval Research Logistics*, vol. 45, no. 7, pp. 733–750, 1998.

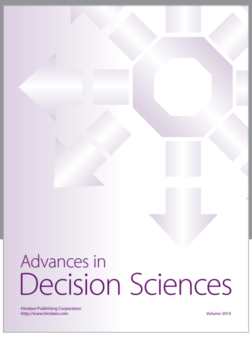
[41] J. Alcaraz and C. Maroto, "A robust genetic algorithm for resource allocation in project scheduling," *Annals of Operations Research*, vol. 102, pp. 83–109, 2001.

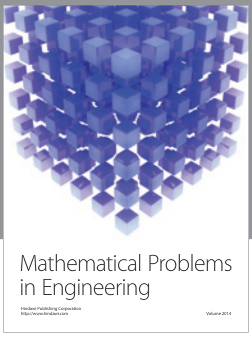
[42] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende, "A genetic algorithm for the resource constrained multi-project scheduling problem," *European Journal of Operational Research*, vol. 189, no. 3, pp. 1171–1190, 2008.

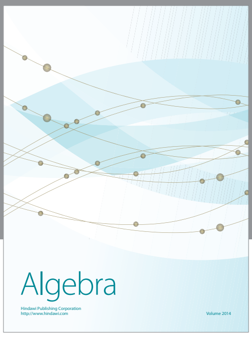
[43] A. A. Najafi and S. T. A. Niaki, “A genetic algorithm for resource investment problem with discounted cash flows,” *Applied Mathematics and Computation*, vol. 183, no. 2, pp. 1057–1070, 2006.

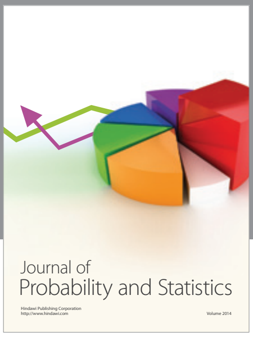
[44] A. A. Najafi, S. T. A. Niaki, and M. Shahsavar, “A parameter-tuned genetic algorithm for the resource investment problem with discounted cash flows and generalized precedence relations,” *Computers and Operations Research*, vol. 36, no. 11, pp. 2994–3001, 2009.


Advances in Operations Research

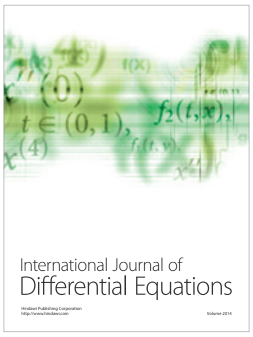

Advances in Decision Sciences


Mathematical Problems in Engineering


Algebra


Journal of Probability and Statistics

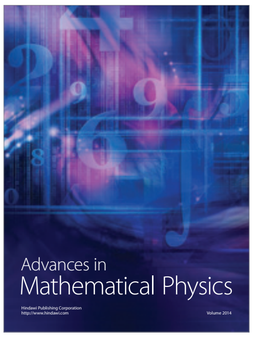
The Scientific World Journal

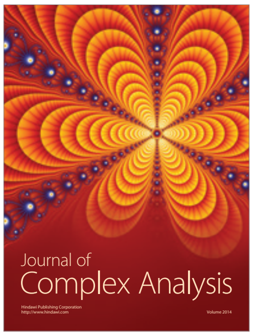

International Journal of Differential Equations

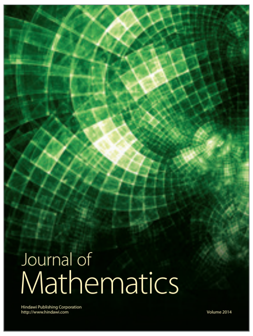
International Journal of Combinatorics

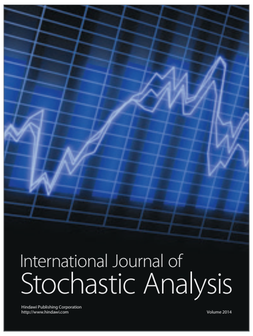
Hindawi

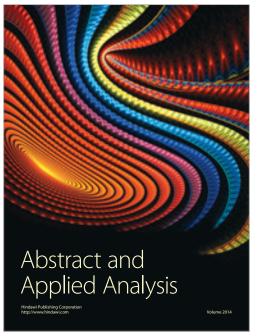
Submit your manuscripts at http://www.hindawi.com

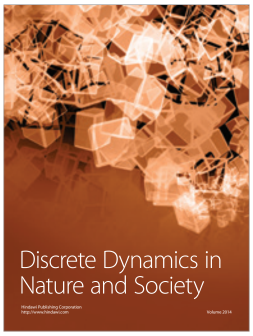

Advances in Mathematical Physics

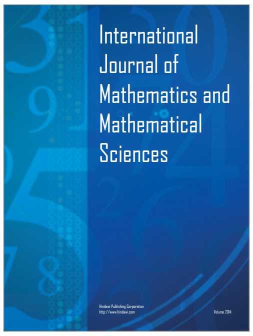

Journal of Complex Analysis


Journal of Mathematics


International Journal of Stochastic Analysis

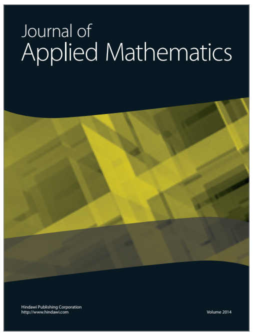

Abstract and Applied Analysis


Discrete Dynamics in Nature and Society


International Journal of Mathematics and Mathematical Sciences

Journal of Discrete Mathematics

Journal of Function Spaces


Journal of Applied Mathematics

Journal of Optimization