Solving multiobjective optimization under bounds by genetic algorithms

Anon Sukstrienwong

***Abstract*—** For complex engineering optimizing problems, several problems are required to be controlled within the specific interval in which something can operate or act efficiently. Most researchers minimize the objective vector into a single objective and interested in the set known as Pareto optimal solution. However, in this paper is concerned with the application of genetic algorithm to solve multi-objective problems in which some objectives are requested to be balanced within its objective bounds. The proposed approach called genetic algorithms for objective boundary (GAsOB scheme) for searching the possible solutions for the particular multi-objectives problems. The elite technique is employed to enhance the efficiency of the algorithm. The experimental results have compared with the results derived by a linear search technique and traditional genetic algorithms through the search space. From the experimental results, GAsOB scheme generates the solution efficiently with customization of the number of eras and immigration rate.

***Keywords*—**Optimization, genetic algorithms, objective boundary, simulation, multiobjective optimization.

I. Introduction

GENTIC algorithms (GAs) have become popular meta-heuristic models for finding solutions to the complex real-life problems. The basic mechanism in GAs is a model of Darwinian evolution. In 1975, Holland had proposed an approach by the use of genetic algorithms [1] which are becoming used increasingly in several fields [2], [4], [5], [25], [27]. Also, examples of using multi-agent systems for solving optimization problems can be found in [22] and [26]. Traditional GAs are customized to a search technique that uses concepts from reproduction and natural selection to produce better offspring for the next generation from existing parents. Only good individuals of the population survive to the next generation while a bad one is eliminated from the selection process. Multiobjective optimization refers to the solution of problems with two or more objectives to be satisfied simultaneously. Often, traditional approaches for solving multi-objective optimization problems try to change the multiple objectives into a single objective problem in which only a global optimal point is desired [6]. A Pareto optimal set is the mathematical solution to a multi-objectives problem [7], [14], [24]. A solution is Pareto-optimal if no other solution can improve one object function without reducing at least one other objectives [9], [12], [19], [20]. However, there exist the multi-objective problems that fail to capture Pareto solutions for multi-objective optimizations [10]. For motivated example, let's assume that $F_k(a)$ and $F_k(b)$ are the values of objective function k of decision vectors a, b $\in$ X. These two parameter vectors are incomparable if and only if they are not equivalent, and neither dominates the other [8]. For equations (3), (4), and (5) shown in this paper, it is hard to say that the objective vector a is better than the objective vector b if both $f_i(a)$ and $f_i(b)$,for all i, are in the requested range of $[l_i, u_i]$. For instance in real-world problems, there exist the problems in which their multi-objective vectors are required to be within the specific bounds. Having good upper and lower bounds is very important in many implicit enumeration methods [23], [25]. For example, in nutritive and health science the researchers try to find good bio-chemical treatments for mental and behavioral disorder patients which make patients recovery effectively with the good health. Of course, the main purpose of the treatments is designed to control the patient's chemical level. However, it is difficult to achieve a low-cost chemical level control.

This paper is an extended work of conference paper by Sukstrienwong [3], so examples and further results are found. The paper' aim is to handle bounded optimization problems. **So** The paper is divided into six sections, including this introduction section. Section 2 provides the fundamental concept of genetic algorithm. Section 3 explains the generality of the genetic algorithm and mathematical definition of multi-objective optimization. Section 4 describes the generality of multi-criteria optimization problems. Then, section 5 shows the proposed algorithm in detail. The simulation setup and experimental results of the scheme are included in this section. The conclusions and future work are in section 6.

II. The fundamental concept of Genetic algorithms

Genetic algorithms are efficient search methods based on principles of natural selection and recombination. Only good individuals in the current population survive to the next generation while a bad one is eliminated from the selection process. The fitness value of each element, which could be the objective of the solution, is used to distinct good and bad individuals from the population. GAs often apply to find the optimal solution to the problem by manipulating a population of solutions. The manners of problems need to be encoded in chromosomes for distinguishing good solutions from bad ones. In general, the way to discriminate the best solutions from bad ones is called the fitness function. Once the problem is encoded in chromosomes with the fixed length, L, the genetic algorithm can be run. There are many variations of encoding schemes of GAs. Often, the individuals are composed of binary (0 or 1) in specified places [9], [21], thus the search space is formed by 2^L points. The genetic algorithm begins with generation 0 with the completely random population.

A. Basic GA-setup

For basic GA-setup, the primary parameters for controlling the genetic algorithm are the population size (*M*) and the maximum number of generations to be run (*Gen*). Secondary parameters, the crossover probability (p_c), and the mutation probability (p_m) are required to create new population. In addition, several other quantitative control parameters and qualitative control variables must be specified in order to completely specify how to execute the genetic algorithm.

In most GA-setup there are two operations to perform during one generation, crossover, and mutation. During the run, a given individual might be mutated, and crossed within a single generation. GA searches the search space of possible population in an attempt to find a good solution based on the fitness value of the chromosomes. The common operators which are used in the GA are described below.

1) Crossover operation

This operator creates two new offspring from two existing parents. Two random parents are selected and recombined with the probability p_c. Suppose that, the crossover probability p_c is 0.95. It means that 95% of current population might participate in crossover as part of creating the next generation. In the past few years, many crossover methods have been designed to apply in different specifications of the problems. The most common crossover techniques are one-point crossover, two-point crossover, and K-point crossover.

- One-point crossover: It is the simplest crossover technique. A single crossover point, which is often between 1 the chromosome length, for both parents is randomly selected. All data beyond that point in either parent are swapped to form two new offspring.
- Two-point crossover: Two random points are selected on the parent strings. Everything between the two points is swapped between the parent, rendering two offspring.
- K-point crossover is an extension technique of one-point crossover and two-point crossover. Originally each point of is chosen randomly. This technique is able to create more offspring at one crossover process.

2) Mutation operation

Basically, Mutation operation operator is used to maintain genetic diversity from the current population for the next generation. This creates a new offspring from an existing member in the population by randomly mutating the character at one position in the chromosome. Mutation probability, p_m, should be relatively low. Suppose p_m is 0.03, that is only 3% of the current population will participate in mutation process for creating the population of the next generation.

III. Multi-objective Optimization formulation

In most real-world problems, several objectives of the problems must be simultaneously satisfied. The common idea for solving these problems is to optimize two or more conflicting objectives to certain constraints. The traditional approach is the combination of all objectives into a single object function. The first multi-objective GA called Vector Evaluated Genetic Algorithm (VEGA) was presented by Schaffer [17]. After that several papers have been published on evolutionary multi-objective optimization including Multi-objective GA [11]. Currently, GAs for multi-criteria are becoming popular for solving practical applications which are required to find the best solution among Pareto optima. In solving multi-objective optimization problems, many traditional methods minimize the objective vector into a single objective, and many researchers may be interested in the set known as Pareto optimal solution [7], [12]. Many papers such as [12], [13], [14], [19], [20] define the definition of Pareto-optimality as following definitions**:**

Definition 1: Consider without loss of generality the following multi-objective optimization problem with an input decision $x = (x_1,\ldots,x_m)$ in the decision space X and an objective $y = (y_1,\ldots,y_m)$ in the objective space Y.

$$\text{Minimize } y = F(x) = (F_1(x_1,\ldots,x_m),\ldots, F_n(x_1,\ldots,x_m)) \qquad (1)$$
$$\text{,where} \quad x = (x_1,\ldots,x_m) \in X$$
$$y = (y_1,\ldots,y_m) \in Y.$$

A decision vector $a \in X$ is said to dominate a decision vector $b \in X$ (also written as $a \succ b$) if and only if:

$$\forall i \in \{1,\ldots,n\}: F_i(a) \geq F_i(b) \text{ and}$$
$$\exists i \in \{1,\ldots,n\}: F_i(a) > F_i(b). \qquad (2)$$

Definition 2: The decision vector $a \in X$ is called Pareto-optimal if and only if a is nondominated regarding the whole decision of X.

The set of all Pareto-optimal points, denote by PS, is called Pareto Set. The set of all the Pareto objective vectors, $PF = \{F(x) \in R^m \mid x \in PS\}$, is called the Pareto front [15]. However, there are some particular problems of which Pareto optimal cannot be applied while considering the problems as multi-objectives. For example, as in this paper, let's assume that $F_i(a)$ and $F_i(b)$ are the values of objective function i (i = 1, …, j) of decision vectors X. In (4) and (5), the objective function i (F_i) is requested to be within the lower limit l_i and upper limit u_i. So, it is unnecessary to find that a dominates or nondominates b if both $F_i(a)$ and $F_i(b)$ are in their boundary.

IV. Problem defined and Proposed Algorithm

A. Problem Defined

The scenario considered in this paper involves the arbitrary multi-objective problem with n decisions. In the particular multi-objective problem, there is a given set K = {k_1, k_2,…, k_n} which is associated with the set of property P = {p_1, p_2,…, p_n}. Moreover each k_q is comprised of j components formed in the vector of (a^q_1, a^q_2,…, a^q_j). Let's an input decision $x_i \in X$, $x_i = (x^i_1, x^i_2, \ldots, x^i_n)$ where x^i_j is an amount of k_j in x_i. Unlike linear programming problems, the objective functions of the problem are divided into two groups which are defined by the following equations:

$$F(x_i) = (F_1(x_i), F_1(x_i), \ldots, F_j(x_i), G(x_i)) \tag{3}$$

First group of objective functions is

$$F_1(x_i) = \frac{(a_1^1 x_1^i + a_1^2 x_2^i + \ldots + a_1^{n1} x_n^i)}{\sum_{k=1}^{n} x_k^i} = \frac{\sum_{k=1}^{n} a_1^k x_k^i}{\sum_{k=1}^{n} x_k^i},$$

$$F_2(x_i) = \frac{(a_2^1 x_1^i + a_2^2 x_2^i + \ldots + a_2^{n1} x_n^i)}{\sum_{k=1}^{n} x_k^i} = \frac{\sum_{k=1}^{n} a_2^k x_k^i}{\sum_{k=1}^{n} x_k^i},$$

…

$$F_j(x_i) = \frac{(a_j^1 x_1^i + a_j^2 x_2^i + \ldots + a_j^{n1} x_n^i)}{\sum_{k=1}^{n} x_k^i} = \frac{\sum_{k=1}^{n} a_j^k x_k^i}{\sum_{k=1}^{n} x_k^i}. \tag{4}$$

And, a second objective function is written as:

$$G(x_i) = \frac{(p_1 x_1^i + p_2 x_2^i + \ldots + p_n x_n^i)}{\sum_{k=1}^{n} x_k^i} = \frac{\sum_{k=1}^{n} p_k x_k^i}{\sum_{k=1}^{n} x_k^i} \tag{5}$$

,where $G(x_i)$ is required to be minimized while $F_k(x_i)$ (k=1, 2, …, j) requested to be balanced within the objective boundary B = {[l_1, u_1], [l_2, u_2], …, [l_j, u_j]}, e. g, [l_1, u_1] is the pair of lower and upper limit of w_1. The objective vectors failed to be in their boundary are undesirable to the solution.

B. *The GAsOB scheme design*

The flowchart of the GAsOB scheme is shown in the Fig. 1. The genetic algorithm begins with generation 0 with the completely random population size M. Note that the GAsOB works on fixed-length strings and searches the space of possible solutions in an attempt to find good decisions based on their fitness value.

1) *Problem encapsulation*

For multi-objective genetic algorithm, let's the chromosomes of the problems are encoded in a sequence of

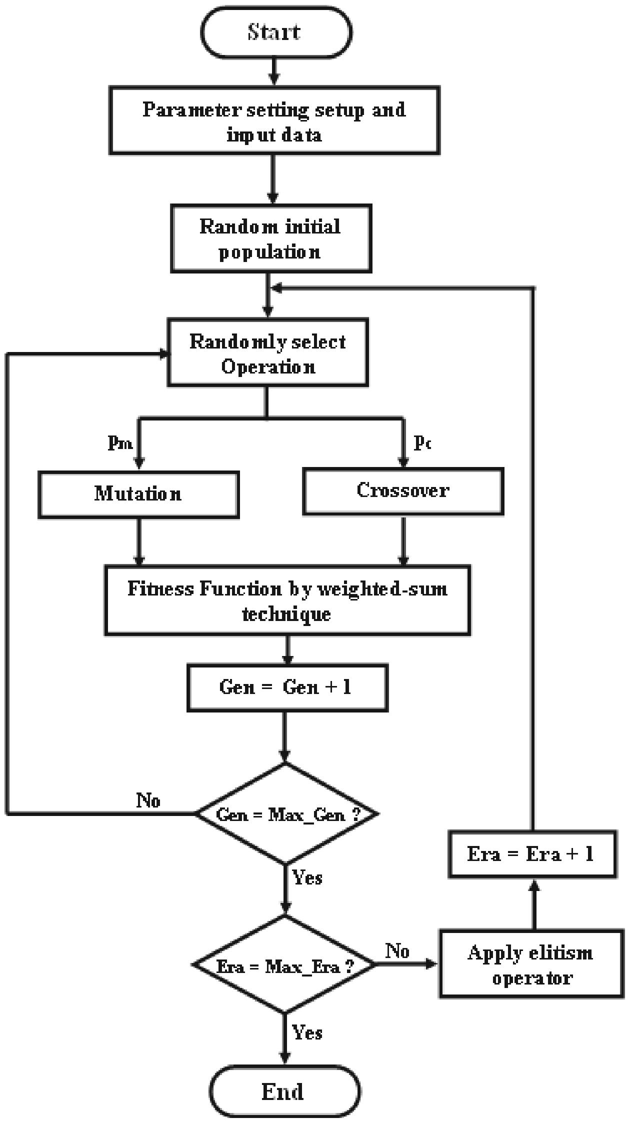

Start
Parameter setting setup and input data
Random initial population
Randomly select Operation
Pm
Pc
Mutation
Crossover
Fitness Function by weighted-sum technique
Gen = Gen + 1
No
Gen = Max_Gen ?
Yes
Era = Era + 1
Era = Max_Era ?
No
Apply elitism operator
Yes
End

Fig. 1 flowchart of GAsOB scheme

$<x^i_1 x^i_2 x^i_3 \ldots x^i_n>$, in which x^i_d represents the amount of k_d, $0 < d \leq n$.

2) *Fitness function*

All functions of (4) and (5) are considered as multi-objective to the problem. However, the functions of (4) are requested to be in the fixed range, objective boundary B = {[l_1, u_1], [l_2, u_2], …, [l_j, u_j]}. The weighted-sum technique is applied to each of $x_i = <x^i_1\ x^i_2\ x^i_3 \ldots x^i_n>$ to obtain a single cost for x_i. The advantage of the weighted sum approach is a straightforward implementation. The weighted sum of x_i can be written mathematical function as follows:

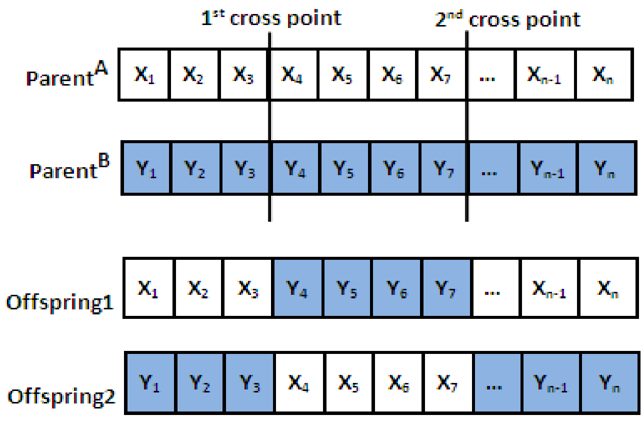
$$w(x_i) = \sum_{j=1}^{n} d_i(x_i)$$

, where $d_j(x_i) = \begin{cases} 1, l_i \le F_j(x_i) \le u_i \\ 0, otherwise \end{cases}$. (6)

For example, let's F_1 and F_2 yield the result as shown in Fig. 2. And, F_1 is requested to be in the range of [0.5, 1.5], while F_2 is requested to be in the range of [1.0, 1.5]. If $F_1(x_i) = 1.0$ and $F_2(x_i) = 0.3$, then $d_1(x_i) = 1$ and $d_2(x_i) = 0$ because only F_1 is in its objective boundary. The weighted sum of two functions of x_i is $w(x_i) = 0+1 = 1$.

3) Parameter setup and GAsOB operators

A set of parameters is initialized in the first era, such as population size (M) of each era, mutation rate (p_m), crossover rate (p_c), immigration rate (Img_rate), maximum generation (Max_Gen), and maximum era (Max_Era). Then, initial population is generated in a complete random. Create a new population for the next generation based on fitness value. Three operators are applied to find solutions shown in Fig. 1.

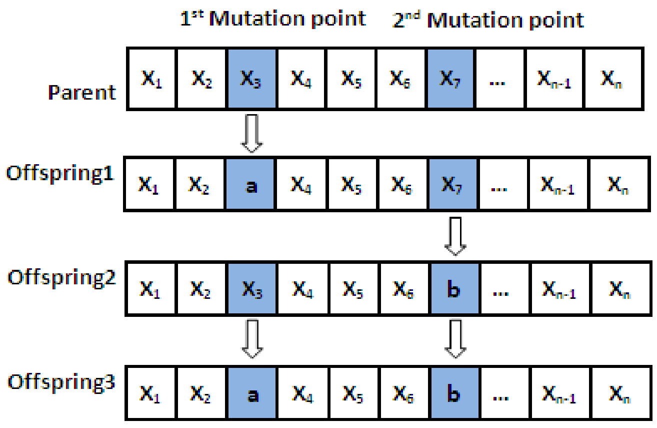

1st cross point
2nd cross point
Parent^A
X1 X2 X3 X4 X5 X6 X7 ... Xn-1 Xn
Parent^B
Y1 Y2 Y3 Y4 Y5 Y6 Y7 ... Yn-1 Yn
Offspring1
X1 X2 X3 Y4 Y5 Y6 Y7 ... Xn-1 Xn
Offspring2
Y1 Y2 Y3 X4 X5 X6 X7 ... Yn-1 Yn
Fig. 3 example of two-point crossover

a) Two-point crossover operator

In this operator, two parents are selected from the matting pool, an intermediate approach to separate only best parents who will produce offspring from the current population. Let's call two parents are x_i and x_j, where i, j ∊ M. Then, two-point crossover is applied by randomly selecting two points on both parents. Everything between the two points is swapped between the parent, rendering two offspring. If the random crossover pointers are 3 and 7, the possible result of two-point crossover is represented in Fig. 3.

b) Two-point mutation operator

Due to the size of chromosomes used for this particular problem, the mutation operation in this approach creates three new offspring from an existing member. The technique is that one random member of the current population is chosen with two mutation points. Suppose x_i is the chosen member and two mutation points are 3 and 7. The possible result of mutation operator is shown in Fig. 4.


1st Mutation point
2nd Mutation point
Parent
X1 X2 X3 X4 X5 X6 X7 ... Xn-1 Xn
Offspring1
X1 X2 a X4 X5 X6 X7 ... Xn-1 Xn
Offspring2
X1 X2 X3 X4 X5 X6 b ... Xn-1 Xn
Offspring3
X1 X2 a X4 X5 X6 b ... Xn-1 Xn
Fig. 4 example of two-point mutation

c) Elitism operator

In [28], [29], elitism has been shown to improve the performance and convergence of the GA in both single and multiple objective applications. In this paper, new offspring created from two-point mutation and two-point crossover are evaluated by the fitness function. Then, new offspring and current parents are sorted by their fitness value. Each individual with bad fitness cost is eliminated, so the rest of the population is transferred to the next generation. It needs to run several generations until it gets the maximum number of generations (Max_Gen). The best members of the current era are found. Reference [6] recommends using the elitism to ensure the best solutions. Some of these winners migrate to the next era. So, in this paper the numbers of immigrating members are controlled by Img_rate. The population of the following era is derived from two parts, selected members of the previous era and its own initial random population with the size of M so that the population size of the following era can be greater than M. In order to avoid that random immigrants disrupt the ongoing search progress too much, the immigrant ratio is usually set to a small value, e.g. 0.2 [16].

V. The Simulation and Experimental Results

A. Simulation Setup

The simulation of the proposed algorithm was implemented more than 3000 lines of Java language on a Pentium (R) D CPU 2.80 GHz, 2 GB of RAM, IBM PC. The simulation has been tried several of runs with different values of the population size (M), mutation probability (p_m), and crossover probability (p_c), to find which values would steer the search towards the best solution. After several runs I have chosen a population size (M) = 400, maximum generation (Max_Gen) for each era = 500, the probability of crossover of 0.95 and the probability of mutation of 0.05. Brief particular set of simulation parameters is shown in Table 1.

Table 1 Summarize the parameter setup of the GAsOB scheme

Variable name	Detail	Value
M	Initial of population size	400
Max_Era	Maximum number of eras	1-10
Max_Gen	Maximum number of generations for each era	500
pc	Crossover probability (p_c)	0.95
pm	Mutation probability (p_m)	0.05
Img_rate	Immigration rate of the current population (%)	0.01-1.0
L	Fixed length of chromosome	14

Table 2 Summarize objective functions and decision parameters.

Constant	Detail	Value
NumOfObjectiveF	Number of objecitve funcitons (F) with their objecitve boundary	6,13
NumOfObjectiveG	Number of objecitve funcitons (G) needed to be minimized	1
NumOfComponent	Maximum number of components of each decision	5,14

B. Data setting

In this paper, two different data sets are used for the GAsOB simulation.

Example 1: Maximization

In this example, the objective functions are bounded within their boundaries listed in Table 3. And, the property and components of each decision are shown in Table 4.

Table 3 the example of Set K and its components used in the simulation

Objectives	Lower bound	Upper bound
$F_1(x_i)$	3500	4200
$F_2(x_i)$	0.10	5.00
$F_3(x_i)$	4.00	-
$F_4(x_i)$	0.01	2.30
$F_5(x_i)$	8.5	-
$Maximize\ G(x_i) = \frac{\sum_{k=1}^{n} x_k^i p_k}{\sum_{k=1}^{n} x_k^i}$		

Table 4 The example of set K and its component used in the simulation

Decision (k_q)	Property (p)	Components				
		a^q_1	a^q_2	a^q_3	a^q_4	a^q_5
k_1	1200	3050	0	1	0.03	7.6
k_2	125	3000	0	3.7	.3	2.5
k_3	25	3000	0.18	10	0.07	12
k_4	101	3900	1.5	2	2	1
k_5	990	5063	2.00	0.9	6.50	58.0

Example 2: Let the multi-objective problems with 13 objective functions are defined as below:

Minimize

$G(x_i)=(5x_1+8.5x_2+8.5x_3+35.78x_4+45x_5+26x_6+24.5x_7+0.31x_8$
$+1.33x_9+ 0.41x_{10}+57x_{11}+80x_{12}+25x_{13})/\sum x_i$

where i = 1 to 13, subject to:

$F_1(x_i) = 0.18x_1+0.35x_5+98x_8+1.28\ x_{13},\ F_1(x_i) \in [400,700]$,

$F_2(x_i) = 10x_1+x_2+2.5x_3+5x_4+2x_5,\ F_2(x_i)\in [1000,3000]$,

$F_3(x_i) = 0.07x_1+0.03x_2+0.03x_3+0.25x_4+0.5x_5+38\ x_8+16\ x_1$, $F_3(x_i) \in [1000, 1300]$,

$F_4(x_i) = 12x_1+7.6x_2+8x_3+38x_4+38x_5+37x_7+94\ x_{10}+58\ x_{11}+73.5\ x_{12},\ F_4(x_i) \in [2000024000]$,

$F_5(x_i) =15.5x_1+1.2x_2+3.6x_3+18x_4+x_5+99x_6+98x_7,\ F_5(x_i)\geq 5000$

$F_6(x_i)=0.45x_1+ 0.27x_2+0.24x_3+2.4x_4+2.55x_5+78.8x_8$, $F_6(x_i)\geq 1500$,

$F_7(x_i)=2900x_1+3500x_2+3300x_3+3750x_4+3370x_5+8800x_6+8000x_7+4250x_8+5280x_{11}+3700x_{12}$, $F_7(x_i)\in[3500000,4000000]$,

$F_8(x_i)=0.25x_1+ 0.04x_2+0.1x_3+0.2x_4+0.42x_5+15.4x_{13}$, $F_7(x_i)\geq 550$,

$F_9(x_i)=1.45x_1+0.18x_2+0.25x_3+0.2x_4+0.6x_5+21x_{13},F_9(x_i) \geq 750$,

$F_{10}(x_i)=0.30x_1+0.30x_2+0.32x_3+1.09x_4+1.11x_5+98.5x_{11}$, $F_{10}(x_i)\geq 900$,

$F_{11}(x_i)=0.20x_1+0.20x_2+0.18x_3+0.57x_5+98.5x_{11},\ F_{11}(x_i)\geq 450$,

$F_{12}(x_i)=0.30x_1+0.28x_2+0.3x_3+1.69x_4+1.51x_5+98.5x_{12}$, $F_{12}(x_i)\geq 910$, and

$F_{13}(x_i)=0.08x_1+0.09x_2+0.07x_3+0.52x_4+0.47x_5,\ F_{13}(x_i) \geq 260$.

C. Experimental results

From the experimental results, there are three factors involving the efficiency of the GAsOB scheme.

1) Immigration rate of elitism operator

As shown in Fig. 5, the maximal value of example 1 derived from the GAsOB scheme is approximately 823 when immigration rate is about 5% of the current population. The best minimal value of example 2 generated by the GAsOB scheme is about 20.5 when immigration is also 5%. This is because the elitism operator immigrants some of the best winners of the current era to the successive era. However, if the immigration rate is over than 5%, the quality of the solution is decreased due to the larger size of the current population.

2) Number of Eras

In both Fig. 7 and 8 the average value derived from GAsOB scheme is convergent to one value when the number of eras (Max_Era) ≥ 5. However, if the Max_Era to be run is big, the computational time of the simulation is also increasing high.

3) Number of Generations for each era

Since the smaller population size helps in reducing the computational time [18], the simulation had tried several numbers of generations to find which numbers tend to generate the best results. When the numbers of the population (M) were increased, the GAsOB generates the better results. The obtained results of both examples are shown in Fig. 9 and Fig. 10. A considerable point in these figures is that when the number of generations was about 300 the quality of GAsOB of all tests is acceptable.

D. Efficiency of the GAsOB scheme

In this section, a new algorithm is also proposed to find the solution as shown in Fig. 11. The algorithm works by searching thought out the search space. It sequentially checks all decision values to see which decision values ($\forall x_i=(k_1,k_2,\ldots,k_m)\in X$) give the best outputs which are within their acceptable range. Then, the algorithm finds the best decision value.

As the simplicity of the algorithm in Fig. 11 is designed, the algorithm is able to find the best answer of the solution if it has one. However, this algorithm takes huge computational time to run when it is encountered with a large number of objective functions. So, only example 1 is able to apply in the algorithm to compare its result with GAsOB scheme. From the experiments, when the number of decisions is 5, the time for running this algorithm is getting slow, about 32 minutes. The results received by GAsOB scheme are almost 96.9% of the algorithm in Fig. 11. The comparison of the GAsOB scheme and the algorithm in Fig. 11 can be summarized in Table 5.

E. The Comparison GAsOB to Traditional GA

In most traditional GA-setup there are two operators to perform during one generation, crossover, and mutation. Briefly justifications for common GA with standard setup are: crossover probability p_c = 0.95, mutation probability p_m = 0.05. From the experience result, the GAsOB scheme gives the best results when Max_Era = 5 and Gen = 200, so the total number of initial population of the GAsOB scheme is 5x400=2000 and the total number of generations is 5x300 = 1500. So, for fair comparison the initial parameters for the traditional GAs are population size (M) = 2000 and maximum generation (Max_Gen) = 1500. And, the fitness function of the traditional GAs is designed as it is appeared in GAsOB scheme. Basically, GAs with the huge population size and high number of generations is possible to give good outputs. However, the computational time is also huge. From experimental results shown in Fig. 12 and Fig. 13, the GAsOB

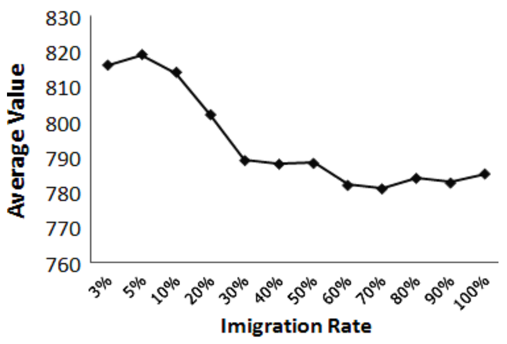


Fig. 5 maximization (example 1)

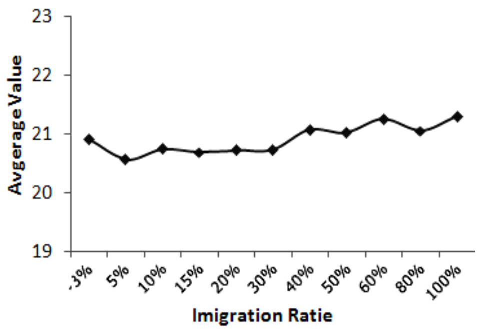


Fig. 6 minimization (example 2)

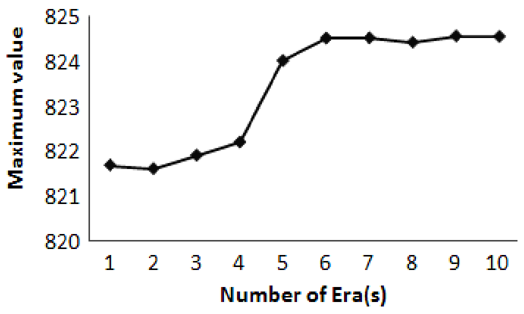


Fig. 7 number of eras example1

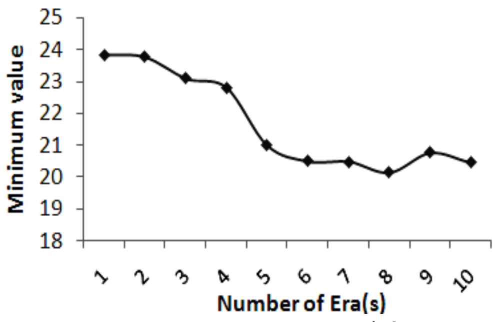


Fig. 8 number of eras example2

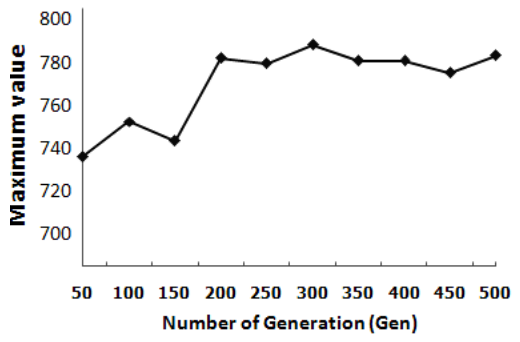


Fig. 9 number of eras example1

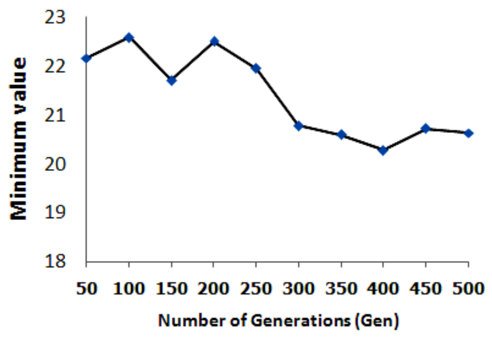


Fig. 10 number of eras example2

```
MinG = 0;
For($0 \le k_1 \le$Max)
  For($0 \le k_2 \le$Max)
    …
      For($0 \le k_{m-1} \le$Max)
        For($0 \le k_m \le$Max)
        {
          Set $x_i = (k_1,k_2,\ldots,k_n)$
          Calculate all objective values which are,
            $F_1(x_i), \ldots, F_n(x_i)$, and $G(x_i)$
          If all of $F_1(x_i),\ldots,F_n(x_i)$ is in its boundary
            If  MinG < $G(x_i)$, then MinG=$G(x_i)$.
        }
```

Fig. 11 an algorithm for finding the solution

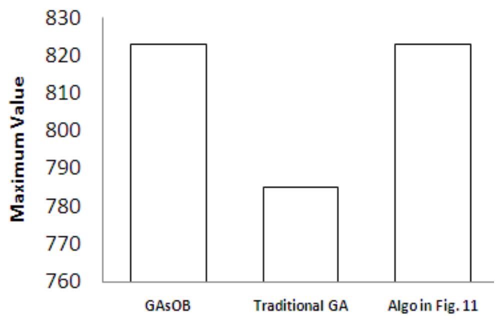


Fig. 12 comparison results to traditional (example 1)

scheme is able to find better solutions than the traditional GA.

Table 5 The comparison results of example 1

Algorithm in Fig. 11		GAsOB	
Computational time	Max $G(x_i)$	Computational time	Max $G(x_i)$
32.23 Min	848.42	3.5 Sec	822.96 (96.9%)

VI. Conclusions and Future Work

In this paper, a method of genetic algorithm called GAsOB has been proposed. The essence of this algorithm is for solving the multi-objective problems in which some objectives are requested to be in certain intervals. From the experiment, there are three factors involved in efficiency of GAsOB scheme; immigration rate, number of eras, and maximum number of generations. The experimental results showed that in most cases the best immigration rate for the GAsOB scheme is approximately 5% of the current population where the initial population (M) equals 300. And, GAsOB scheme needs at least 5 eras to get the best results. The experiment shows that the proposed algorithm is able to find the solution about 96.9% of the accuracy value. The computational time for GAsOB to search for the optimal solution is acceptable. For future works, the proposed algorithm will investigate in complex real-world problems and compare with other well-known approaches.

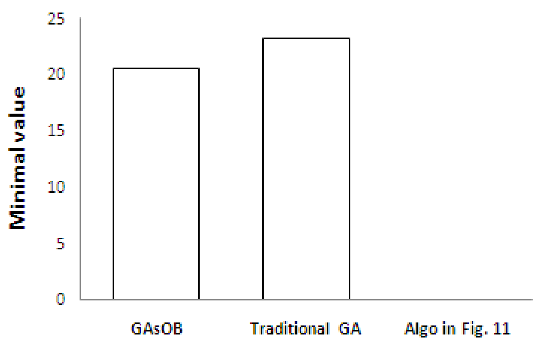


Fig. 13 comparison results to traditional (example 2)

References

[1] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, Michigan, USA.
[2] A. R Awad and I. Von Poser, Calibrating Conceptual Rainfall-Runoff Models Using a Real Genetic Algorithm Combined with a Local Search Method, Latest trends on computers, 14th WSEAS International Conference on COMPUTERS, Corfu Island, Greece, July 2010, WSEAS, WSEAS Press, Vol. I, pp. 174-181.
[3] A. Sukstrienwong. Genetic Algorithm for Multi-Objectives Problems under its Objective Boundary., Selected Topics in Applied Computer Science, 10th WSEAS International Conference on Applied Science, Iwate Prefectural University, Japan, October 2010, WSEAS, WSEAS Press, pp. 38-43.

[4] S. Sornmuang and S. Sujitjorn, GA-Based PIDA Control Design Optimization with an Application to AC Motor Speed Control, International Journal of Mathematics and Computers in Simulation , Issue 3, Volume 4, 2010, NUAN, pp. 67-80.
[5] M. Yoshikawa, H. Nishimura, and H. Terai, A New Genetic Coding for Job Shop Scheduling Problem Considering Geno Type and Pheno Type, Recent Advances in Computer Engineering and Applications, Proc. of the 4th WSEAS International Conference on Computer Engineering and Applications, Harvard University, Cambridge, January 27-29, 2010, WSEAS, WSEAS Press, pp 59-62.
[6] Hamidreza Eskandari, Luis Rabelo and Mansooreh Mollaghasemi. Multiobjective Simulation Optimization Using an Enhanced Genetic Algorithm.
[7] N.Suguna, and K.Thnushkodi., Genetic Algorithm Based Feature Ranking in Multi-criteria Optimization. In: IJCSNS international Journal of Computer Sciences and Network Security, Vol. 9, No. 6, pp. 132-140.
[8] S. Huband, P.Hingston, L. Barone, and L. While, A Review of Multiobjective Test Problems and a scalable Test Problem Toolkit. In: IEEE-Transactions on Evolutionary Computation. Vol. 10, No. 5, October 2006.
[9] J. Lohn, W. Kraus, G. Haith., Comparing a Coevolutionary Genetic Algorithm for Multiobjective Optimization, Proceedings of the 2002 IEEE Congress on Evolutionary Computation, May 2002, pp. 115-1162.
[10] Messac, A., Sundaraj, G. J., Tappeta, R. V., and Renaud, J. E. Ability of Objective Functions to Generate Points on Non-convex Pareto Frontiers. AIAA Journal, Vol. 38, No. 6, June 2000, pp. 1084-1091.
[11] Fonseca CM, Fleming PJ. Multiobjective Genetic Algorithm. In: IEE colloquium on 'Genetic Algorithms for Control Systems Engineering' (Digest No. 1993/130), 28 May 1993. London, UK: IEE; 1993.
[12] N.Srinvas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. The Journal of Evolutionary Computation, Vol. 2, No. 3, 1994, pp 221-248.
[13] Ivo F. Sbalzarini, Sibylle Miiller, and Petros Koumoutsakos., Multiobjective optimization using evolutionary algorithms. Proceedings of the Summer Program 2000.
[14] Zitzler, E. and L. Thiele (1999).Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271.
[15] Hui Li and Qingfu Zhang. Multiobjective Optimization Problems with Complicated Pareto, MOEA/D and NSGA-II. In: IEEE Transactions on Evolutionary Computation. Vol. 13, No. 2. April 2009.
[16] Shengxiang Yang. Genetic Algorithm with Elitism-Based Immigrants for Changing Optimization Problems. Evolutionary Computation Journal. Vol. 16, No. 3, pp. 385-416, 2008.
[17] Schaffer JD. Multiple Objective Optimization with Vector Evaluated Genetic Algorithm. In: Proceedings of the international conference on genetic algorithm and their applications, 1985.
[18] Shisanu Tongchim and Prabhas Chongstitvatana.. Parallel genetic algorithm with parameter adaptation.
[19] Antony W. Iorio and Xiaodong Li. Solving rotated multi-objective optimization problems using differential evolution. In AI 2004: Advances in Artificial Intelligence, Proceedings, pages 861–872. Springer- Verlag, Lecture Notes in Artificial Intelligence Vol. 3339, 2004.
[20] Luis Vicente Santana-Quintero and Carlos A. Coello Coello. An Algorithm Based on Differential Evolution for Multi-Objective Problems. International Journal of Computational Intelligence Research, 1(2):151–169, 2005.
[21] E. Mezura-Montes, M. Reyes-Sierra and C. A. Coello Coello: Multi-objective optimization using differential evolution: a survey of the state-of-art, Advances in Differential Evolution, Springer, 2008, pp. 173–196.
[22] Ohbyung Kwon, Ghiyoung Im, and Kun Chang. Mace-scm: An effective supply chain decision making approach based on multi-agent and case-based reasoning. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, 2005.
[23] C. Delort and O. Spanjaard. Using bound sets in multiobjective optimization: Application to the biobjective binary knapsack problem. In 9th International Symposium on Experimental Algorithms (SEA 2010), volume 6049 of Lecture Notes in Computer Science, pages 253–265, 2010.
[24] K. Harada, J. Sakuma, K. Ikeda, I. Ono, and S. Kobayashi. Local search for multiobjective function optimization: Pareto descent method. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006), pages 659–666, New York, NY, 2006. ACM Press.
[25] Andre A. Keller. Genetic Search Algorithm to Fuzzy Multiobjective Games: a Mathematica Implementation for Multi-Objectives Problems under its Objective Boundary., Selected Topics in Applied Computer Science, 10th WSEAS International Conference on Applied Science, Iwate Prefectural University, Japan, October 2010, WSEAS, WSEAS Press, pp. 351-359.
[26] Qiang Wei, Tetsuo Sawaragi, and Yajie Tian. Bounded optimization of resource allocation among multiple agents using an organizational decision model. Advanced Engineering Informatics, 19:67–78, 2005.
[27] K. Tagawa and H. Kim: A local search technique for large-scale optimum design problems of double mode SAWfilters, WSEAS Trans. on Circuits and Systems, Issue 1, Vol. 6, 2007, pp. 1–8.
[28] P. M. Reed, B. S. Minsker, and D. E. Goldberg, "The practitioner's role in competent search and optimization using genetic algorithms," presented at the World Water and Environmental.
[29] C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithms," IEEE Trans. Evol. Comput., vol. 7, no. 4, pp. 367–385, Aug. 2003.

Anon Sukstrienwong received the B.Sc. in applied mathematics from King Mongkut's Institute of Technology Ladkrabang, Thailand in 1992. He received his master's degree in engineering from the University of Colorado, Denver, U.S.A. in 1994. He is currently an assistant professor in the department of computer and technology at Bangkok University in Bangkok, Thailand.
His current research interests include: evolutionary multiobjective optimization, evolutionary algorithms in general.