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

GENETIC 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 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 $f_i(a)$ and $f_i(b)$ 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’s 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 $2^k$ 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 a 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 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 objective 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 $x = (x_1, ..., x_n)$ in the decision space $X$ and an objective $y = (y_1, ..., y_m)$ in the objective space $Y$.

Minimize $y = F(x) = (F_1(x), ..., F_m(x))$ \hspace{1cm} (1)

\[ \text{where } x = (x_1, ..., x_n) \in X \]
\[ y = (y_1, ..., y_m) \in Y. \]

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:

\[ \forall i \in \{1, ..., n\}: F_i(a) \geq F_i(b) \text{ and } \exists i \in \{1, ..., n\}: F_i(a) > F_i(b). \hspace{1cm} (2) \]

**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 $P$, is called Pareto Set. The set of all the Pareto objective vectors, $PF = \{F(x) \in R^m | x \in P\}$, 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 $K = \{k_1, k_2, \ldots, k_n\}$ which is associated with the set of property $P = \{p_1, p_2, \ldots, p_n\}$. Moreover each $k_q$ is comprised of j components formed in the vector of $(a_{q1}, a_{q2}, \ldots, a_{jq})$. Let’s an input decision $x_i \in X$, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ where $x_{ij}$ 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:

$$F(x_i) = (F_1(x_i), F_2(x_i), \ldots, F_j(x_i), G(x_i))$$ (3)

First group of objective functions is

$$F_1(x_i) = \frac{\sum_{k=1}^{n} a_{k1} x_{ik}^1 + a_{k2} x_{ik}^2 + \ldots + a_{kn} x_{ik}^n}{\sum_{k=1}^{n} x_{ik}^k},$$

$$F_2(x_i) = \frac{\sum_{k=1}^{n} a_{k1} x_{ik}^1 + a_{k2} x_{ik}^2 + \ldots + a_{kn} x_{ik}^n}{\sum_{k=1}^{n} x_{ik}^k},$$

...  

$$F_j(x_i) = \frac{\sum_{k=1}^{n} a_{k1} x_{ik}^1 + a_{k2} x_{ik}^2 + \ldots + a_{kn} x_{ik}^n}{\sum_{k=1}^{n} x_{ik}^k}. (4)$$

And, a second objective function is written as:

$$G(x_i) = \frac{\sum_{k=1}^{n} p_{k1} x_{ik}^1 + p_{k2} x_{ik}^2 + \ldots + p_{kn} x_{ik}^n}{\sum_{k=1}^{n} x_{ik}^k}. (5)$$

where $G(x_i)$ is required to be minimized while $F_k(x_i)$ $(k=1, 2, \ldots, j)$ requested to be balanced within the objective boundary $B = \{[l_1, u_1], [l_2, u_2], \ldots, [l_j, u_j]\}$, e. g, $[l_i, u_i]$ is the pair of lower and upper limit of $w_i$. 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

$$<x_{i1}^1 x_{i2}^1 \ldots x_{in}^1>,$$

and $w_i^d$ 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 $B = \{[l_1, u_1], [l_2, u_2], \ldots, [l_j, u_j]\}$ The weighted-sum technique is applied to each of $x_i = <x_{i1}^1 x_{i2}^1 \ldots x_{in}^1>$ 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:

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

Fig. 1 flowchart of GAsOB scheme
where $d_j(x_i) = \begin{cases} 1, & i_j \leq F_j(x_i) \leq u_i \\ 0, & \text{otherwise} \end{cases}$.

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 ($\text{Img} \_ \text{rate}$), maximum generation ($\text{Max} \_ \text{Gen}$), and maximum era ($\text{Max} \_ \text{Era}$). Then, initial population is generated in a completely random. Create a new population for the next generation based on fitness value. Three operators are applied to find solutions shown in Fig. 1.

a) Two-point crossover operator

In this operator, two parents are selected from the mating 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 $i, j \in M$. 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.

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 ($\text{Max} \_ \text{Gen}$). 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 $\text{Img} \_ \text{rate}$. 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 ($\text{Max} \_ \text{Gen}$) 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

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

Table 2 Summarize objective functions and decision parameters.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Detail</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumOfObjectiveF</td>
<td>Number of objective functions (F) with their objective boundary</td>
<td>6,13</td>
</tr>
<tr>
<td>NumOfObjectiveG</td>
<td>Number of objective functions (G) needed to be minimized</td>
<td>1</td>
</tr>
<tr>
<td>NumOfComponent</td>
<td>Maximum number of components of each decision</td>
<td>5,14</td>
</tr>
</tbody>
</table>

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.

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

Minimize
\[ G(x_i) = (5x_1 + 8.5x_2 + 8.5x_3 + 35.78x_4 + 45x_5 + 26x_6 + 24.5x_7 + 0.31x_8 + 1.33x_9 + 0.41x_10 + 57x_11 + 80x_12 + 25x_13) / \sum_i x_i \]

where \( i = 1 \) to 13, subject to:
\[ F_1(x_i) = 0.18x_i + 0.35x_i + 9.8x_i + 1.28x_i, \quad F_1(x_i) \in [400,700], \]
\[ F_2(x_i) = 10x_i + x_i + 2.5x_i + 5x_i + 2x_i, \quad F_2(x_i) \in [100,3000], \]
\[ F_3(x_i) = 0.07x_i + 0.03x_i + 0.03x_i + 0.25x_i + 0.5x_i + 0.5x_i + 0.5x_i + 38x_i + 16x_i, \]
\[ F_4(x_i) \in [1000,1300], \]
\[ F_4(x_i) = 12x_i + 7.6x_i + 8x_i + 38x_i + 38x_i + 37x_i + 94x_i + 58x_i + 73.5x_i + 12x_i, \]
\[ F_4(x_i) \in [20002400], \]
\[ F_5(x_i) = 15.5x_i + 1.2x_i + 3.6x_i + 18x_i + 1x_i + 99x_i + 98x_i, \]
\[ F_5(x_i) \geq 5000 \]
\[ F_6(x_i) = 0.45x_i + 0.27x_i + 0.24x_i + 2.4x_i + 2.5x_i + 78.8x_i, \]
\[ F_6(x_i) \geq 15000 \]
\[ F_7(x_i) = 2900x_i + 3500x_i + 3300x_i + 3750x_i + 3370x_i + 8800x_i + 8000x_i + 4250x_i + 5280x_i + 3700x_i + 12, \]
\[ F_7(x_i) \in [3500000, 4000000], \]
\[ F_8(x_i) = 0.25x_i + 0.04x_i + 0.1x_i + 0.2x_i + 0.42x_i + 0.15x_i + 13, \]
\[ F_8(x_i) \geq 550, \]
\[ F_9(x_i) = 1.45x_i + 0.18x_i + 0.25x_i + 0.2x_i + 0.6x_i + 21x_i, \]
\[ F_9(x_i) \geq 750, \]
\[ F_10(x_i) = 0.3x_i + 0.3x_i + 0.32x_i + 1.09x_i + 1.11x_i + 98.5x_i, \]
\[ F_10(x_i) \geq 900, \]
\[ F_{11}(x_i) = 0.20x_i + 0.20x_i + 0.18x_i + 0.57x_i + 98.5x_i, \]
\[ F_{11}(x_i) \geq 450, \]
\[ F_{12}(x_i) = 0.3x_i + 0.28x_i + 0.3x_i + 1.69x_i + 51.5x_i + 98.5x_i, \]
\[ F_{12}(x_i) \geq 910, \]
\[ F_{13}(x_i) = 0.08x_i + 0.09x_i + 0.07x_i + 0.52x_i + 0.47x_i, \]
\[ F_{13}(x_i) \geq 260. \]

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 throughout the search space. It sequentially checks all decision values to see which decision values (∀x=(k₁,k₂,…,kₘ)∈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 pₙ = 0.95, mutation probability pₜ = 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
scheme is able to find better solutions than the traditional GA.

Table 5 The comparison results of example 1

<table>
<thead>
<tr>
<th>Algorithm in Fig. 11</th>
<th>GAsOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational time</td>
<td>Max G(x)</td>
</tr>
<tr>
<td>32.23 Min</td>
<td>848.42</td>
</tr>
</tbody>
</table>

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.

REFERENCES


Shisana Tongchim and Prabhas Chongstitvatana.. Parallel genetic algorithm with parameter adaptation.


