Simple Addition of Ranking Method for Constrained Optimization in Evolutionary Algorithms

Pei Yee Ho
Department of Bioscience and Bioinformatics
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, Fukuoka
820-8502, Japan
peiyee@yahoo.com

Kazuyuki Shimizu*
Department of Bioscience and Bioinformatics
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, Fukuoka
820-8502, Japan
shimi@bio.kyutech.ac.jp

ABSTRACT

During the optimization of a constrained problem using evolutionary algorithms (EAs), an individual in the population can be described using three important properties, i.e., objective function, the sum of squares of the constraint violation, and the number of constraints violated. However, the question of how to combine these three properties effectively is always difficult to solve due to the scaling and aggregation problems. In this paper, a simple addition of ranking method is proposed to handle constrained optimization problems in EAs. In this method, each individual is ranked based on the above three properties separately, resulting in three new properties which are in the same order of magnitude. Simple addition of the three new terms can then be performed and this produces a new global ranking for each individual. The algorithm was tested using 13 benchmark problems on the basis of evolution strategy and genetic algorithm. Results showed that the proposed algorithm performed well in all of the problems with inequality constraints, without requiring any parameter tuning for the constraint handling part. On the other hand, problems with equality constraints can be handled well through the addition of a simple diversity mechanism and a tolerance value adjustment scheme.

Categories and Subject Descriptors: G.1.6 [Numerical Analysis]: Optimization — *constrained optimization, global optimization*

General Terms: Algorithms, Experimentation, Performance

Keywords: Constraint handling, evolutionary algorithms, simple ranking, single objective optimization

*The author is also affiliated to Institute for Advanced Biosciences, Keio University, Tsuruoka, Yamagata, 997-0017, Japan.

GECCO'05, June 25–29, 2005, Washington, DC, USA.

1. INTRODUCTION

Consider a general nonlinear optimization problem formulated as minimizing $f(\vec{x})$, where $\vec{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$, subject to $g_j(\vec{x}) \leq 0$, $j = 1, \ldots, m$, $h_j(\vec{x}) = 0$, $j = m+1, \ldots, p$, and bounded by $x_i^l \leq x_i \leq x_i^u$, $i = 1, \ldots, n$. Both inequality constraints $g_j(\vec{x})$ and equality constraints $h_j(\vec{x})$ can be in the form of linear or nonlinear equations. For equality constraints, they can be transformed into inequality constraints through the use of a very small tolerance value δ, as described by

$$g_j(\vec{x}) = |h_j(\vec{x})| - \delta \leq 0\,, \quad j = m+1, \ldots, p\ . \qquad (1)$$

To solve the above optimization problem, EAs have been widely adopted due to their flexibility and adaptability to the task in hand, in combination with the robust performance and global search characteristics [2]. However, EAs are unconstrained optimization methods which require an addition of constraint handling part to make them useful in real world applications. A recent review by Coello Coello [4] has discussed on various techniques available for the handling of constraints in EAs. Other than that, recently published new techniques included the use of an improved diversity mechanism [7], the use of multiobjective optimization concept with a new ranking scheme [1] or a mechanism to reduce the constrained search space [5]. Despite the varieties available, the main issues when dealing with constraint handling in EAs are concerned with the scaling and aggregation of objective function and constraint violation, and also on the maintenance of diversity within a population such that the global minimum can be found.

In this paper, a simple addition of ranking method is proposed. Although the idea of individuals ranking based on objective function and constraint violation is not new, e.g., the works of Angantyr et al. [1], Ray et al. [8], and Runarsson and Yao [10], the proposed method offers a much simpler way of ranking and aggregation strategies when compared to the existing methodologies. For the optimization problems with equality constraints, a simple reinitialization scheme of the whole population is incorporated to maintain diversity within the population. To ease the search of feasible individuals within the search space, a tolerance value adjustment scheme is also incorporated. It is the aim of this paper to show that the concept of ranking addition is able to solve highly constrained optimization problems as effective as the existing approaches, but with a much simpler algorithm.

Table 1: An example of ranking assignment for a population with 6 individuals

Individual	Feasible	$f(\vec{x})$	R_{f}	s	R_{s}	v	R_{v}
Ind 1	Yes	10	4	0	1	0	1
Ind 2	No	20	5	50	6	3	6
Ind 3	No	-1	2	30	5	1	3
Ind 4	Yes	500	6	0	1	0	1
Ind 5	No	-20	1	10	3	1	3
Ind 6	No	0	3	20	4	2	5

The paper is organized as follows: Section 2 presents the general ideas of the proposed methodology. Section 3 gives the complete algorithm in the basis of evolution strategy (ES) and genetic algorithm (GA). Section 4 shows the results on the studies of 13 benchmark problems, and finally Sect. 5 concludes the paper.

2. GENERAL IDEAS

Considering an individual from a population of any EAs, it can be easily described in terms of three important numerical properties, i.e., (1) $f(\vec{x})$, the value of objective function; (2) $s = \sum_{j=1}^{p} \max[0, g_j(\vec{x})]^2$, the sum of squares of constraint violation; and (3) v, the number of constraints violated ($0 \leq v \leq p$). Since these three properties are in different orders of magnitude, the question now would be *"How can we combine these three properties into ONE term?"*

To solve this question, a simple ranking method is used. By comparing an individual against all other members in the population, the individual can be ranked in terms of $f(\vec{x})$, s, and v, respectively, which produces three new terms denoted by R_{f}, R_{s}, and R_{v}. In here, each individual is assigned a rank equals to one plus the number of individuals which dominate it. To make it clearer, an example of ranking assignment for a population with 6 individuals is shown in Table 1. It can be clearly seen from this table that the three new terms R_{f}, R_{s}, and R_{v} are in the same order of magnitude. As a result, mathematical manipulation of these three terms can be performed easily without bias.

After solving the scaling problem, the next issue is the aggregation problem. The following aggregation strategy is proposed:

Condition 1. All of the individuals in the population are infeasible,

$$\phi(\vec{x}) = R_{\mathrm{s}} + R_{\mathrm{v}} \tag{2}$$

Condition 2. Feasible individuals exist in the population,

$$\phi(\vec{x}) = R_{\mathrm{f}} + R_{\mathrm{s}} + R_{\mathrm{v}} \tag{3}$$

In here, $\phi(\vec{x})$ is the new global ranking for each individual, and this function then becomes the new objective function to be minimized. Equation (2) is proposed based on the fact that when all of the individuals of a population are infeasible, the information from $f(\vec{x})$ is not important. When there is a mixture of feasible and infeasible individuals, both of the information from objective function and constraint violation become important. The concept of simple addition of ranking in (3) is proposed based on the fact that an infeasible individual with only slight constraint violation and small f value, is considered as better than another individual which is feasible but with a much larger f value. For example, the infeasible Individual 5 in Table 1 ($\phi(\vec{x}) = 7$) is more preferable than the feasible Individual 4 which has a much larger f value ($\phi(\vec{x}) = 8$).

Besides scaling and aggregation, there are another two difficulties when dealing with equality constraints. First, with a very small tolerance value (e.g., $\delta = 0.0001$), searching for the resulting extremely small feasible regions becomes difficult [6]. Second, once the first feasible individual is found, further minimization in the objective function is difficult because feasible regions are normally disjoint among each other. As a result, the optimization is trapped inside a local minimum and unable to find the global minimum.

Inspired by the tolerance value adaptation scheme from Hinterding [6], the same concept is used but with a different approach. Instead of adapting the δ value based on a certain feedback, the optimization is run by changing the δ value in (1) alternatively between two different values, δ_1 and δ_2. For an optimization problem with a total number of p constraints where m of them are inequality constraints, we have $\delta_{1j} = A_j$, where $A_j \gg \delta_{2j}$, and $\delta_{2j} = 0.0001$, for $j = m+1, \ldots, p$. In this paper, δ_{1j} and δ_{2j} will be used interchangeably with the notation of δ_1 and δ_2 respectively for the ease of readability. When the optimization is run using δ_2, the solutions found are feasible and these solutions are considered to be located in the *feasible regions*. The feasible region that contains the optimal solution is named as *target region* in here. On the other hand, the optimization using δ_1 enables us to find solutions which are located near to the feasible regions. We define the regions containing such solutions as *search regions*.

To find a suitable value for δ_{1j}, we use the information from (1). By ignoring the δ value in (1), we have the maximum value of the function as $\max(|h_j(\vec{x})|)$. We define

$$\delta_{1j} = B \times \max(|h_j(\vec{x})|), \quad j = m+1, \ldots, p, \tag{4}$$

where B can be varied from 0 to 1. The smaller the value of B, the closer is the solution to the feasible region. The use of $\max(|h_j(\vec{x})|)$ in (4) helps in determining an upper bound for δ_{1j} and this is necessary because $h_j(\vec{x})$ can be in different orders of magnitude for $j = m+1, \ldots, p$.

After deciding the values for δ_1, we need to define the criteria so that the algorithm can shift within δ_1 and δ_2 effectively. By using the new objective functions given by (2) and (3), the following algorithm is proposed:

1. Start the optimization using δ_1 (by replacing δ in (1) with δ_{1j})
2. After the first feasible individual that fulfills δ_1 is found, continue with the optimization for k generations
3. Continue with the optimization using δ_2 (by replacing δ in (1) with δ_{2j})
4. Once the first feasible individual that fulfills δ_2 is found, stop the optimization, reinitialize the whole population, go back to Step 1 again

By letting the algorithm to run for k generations using δ_1 as the tolerance value, the objective function value can be further minimized before it enters the feasible region (δ_2 region). However, the optimization is stopped once the first feasible individual is found in the δ_2 region. This is because the jumping from one local minimum to another is normally difficult due to the disjoint between two feasible regions.

Thus, continuing minimizing in this local minimum becomes meaningless in some cases. To maintain diversity within the population, reinitialization of the whole population is performed in Step 4. Through the use of different initial starting points, the chances of finding the global minimum become higher. In short, we have two parameters, B and k, to be tuned for the problems with equality constraints.

3. ALGORITHMS

To incorporate the ideas from Sect. 2, the (μ, λ)-ES as adopted by Runarsson and Yao [9] is used. This ES is used due to its simplicity. The exactly similar parameter settings as that in [9] are also used such that a fairer comparison of results can be made.

A brief description of the ES algorithm is given below (details please refer to [9]). In the (μ, λ)-ES, each individual i is described by $(\vec{x}_i, \vec{\sigma}_i)$, $i = 1, \ldots, \lambda$, where $\vec{x}$ is the vector of objective variables $(x_1, \ldots, x_n)$, and $\vec{\sigma}$ is the vector of mean step sizes for mutation (strategy parameters) $(\sigma_1, \ldots, \sigma_n)$. The initial population of $\vec{x}$ is generated using uniformly distributed random numbers across the variable bounds. On the other hand, the initial setting of the mean step size is $\sigma_{i,j}^{(0)} = \left(x_j^u - x_j^l\right)/\sqrt{n}, i = 1, \ldots, \lambda, j = 1, \ldots, n$. This initial value is then used as the upper bound for $\vec{\sigma}$.

After evaluating the values of $f(\vec{x})$, s, and v for each individual, we can then easily rank all of the individuals based on these three values respectively and obtain R_{f}, R_{s}, and R_{v}. Using (2) and (3), we get the new global ranking $\phi(\vec{x})$ for each individual. By sorting the population with respect to $\phi(\vec{x})$, the best μ individuals (with the smallest $\phi(\vec{x})$) out of λ are selected to become the parents for the next generation. The (μ, λ) setting used is the same as that in [9], i.e., $(30, 200)$-ES. The parents are then used to create λ/μ offspring on average.

The strategy parameters corresponding to the selected μ individuals are used to generate λ new strategy parameters through replication. To vary the strategy parameters, global intermediate recombination between two parents and adaptation through log-normal update rule as described in [9] are implemented. The term expected rate of convergence (φ^*) used in the update rule is set to 1. The new λ strategy parameters are then used to mutate the offspring. When an offspring created is outside the variable bounds, mutation will be retried for a maximum of ten times (following [9]). Figure 1(a) shows the general flow chart of the algorithm. By following Steps 1 to 4 in Sect. 2 and denoting the steps within the dash box of Fig. 1(a) as Algorithm P1, we obtain the general flow chart of the algorithm for problems with equality constraints as illustrated in Fig. 1(b).

Besides ES, GA is another type of EAs that has been widely used. When compared with ES, GA is much more complicated with the large varieties in the fitness assignment, selection, recombination, and mutation schemes. The reason of using GA as the basis for evaluation is to show that the proposed methodology can work well in different types of EAs. The general flow chart of the algorithm is shown in Fig. 2. Since some of the parents from the population will be retained to the next generation when dealing with GA or $(\mu + \lambda)$-ES, the ranking procedure needs to be performed twice instead of once as that in (μ, λ)-ES. This is because ranking process has to be performed with respect to the whole population involved, i.e., parents + offspring.

The corresponding algorithm for the problems with equality constraints is similar to that shown in Fig. 1(b) with only one exception, i.e., only the variables $\vec{x}$ needs to be reinitialized in the case of GA. This is different from the ES in which both $\vec{x}$ and $\vec{\sigma}$ are reinitialized at the same time. This is due to the difference in the methods of generating mutation step sizes in both algorithms.

The GA from Chipperfield et al. [3] is adopted in this study. The following operator schemes are used: population representation and initialization - real value; fitness assignment - linear ranking-based (selective pressure = 2); selection function - stochastic universal sampling; crossover operator - intermediate recombination; mutation operator - real value mutation with no shrinking of mutation range; reinsertion - fitness-based; migration - fitness-based migration using complete net structure. Other parameters included: number of subpopulations = 4; number of individuals per subpopulation = 50; generation gap = 0.95; crossover rate = 0.7; mutation rate = $1/n$; insertion rate = 1; migration rate = 0.2; and number of generations between migration = 20. The number of individuals used here is $4 \times 50 = 200$, which is the same as that of ES. Other parameter settings chosen here are sort of typical values used in GA.

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

To evaluate the performance of the proposed methodology, thirteen benchmark problems as that used in [7] and [9] were tested. A summary of the characteristics of the optimization problems is shown in Table 2, where LI and NI are the numbers of linear and nonlinear inequality constraints, and LE and NE are the numbers of linear and nonlinear equality constraints. These test problems are representative of difficult global optimization problems and the degree of difficulties indicated by the ratio of feasible region to that of entire search space can be found in [7]. In here, maximization problems are converted to minimization using $-f(\vec{x})$. For each test, 100 independent runs were performed. All of the runs were terminated after 1,750 generations, with the exception of 175 generations for problem `g12`, similar to that used in [9].

The first test is to check the ability of (2) to find the first feasible individual from the search space. We run the test by using the algorithm in Fig. 1(a). In here, δ value is set to 0.0001 for `g03`, `g05`, `g11`, and `g13`. As a comparison, the test was also run by replacing (2) with $\phi(\vec{x}) = R_{\mathrm{s}}$. The purpose is to see whether R_{v} is important in finding the feasible individuals. Table 2 shows the number of independent runs where feasible individuals can be found (N_{s}), and also the median of generation number when the first feasible individual is found (G_{fea}) with its corresponding standard deviation (Std), based on a total of 100 independent runs.

From Table 2, we can see that feasible individuals could be found in all of the runs when using $\phi(\vec{x}) = R_{\mathrm{s}} + R_{\mathrm{v}}$. The results were similar when using $\phi(\vec{x}) = R_{\mathrm{s}}$, except in `g05`. This shows that although the information of s alone is enough to find the feasible individuals, the number of constraints violated (v) can help in improving the search of feasible individuals for problems which is highly constrained. From the column of G_{fea}, we can see that the algorithm was able to find the feasible individuals in less than 30 generations for all of the problems, except in `g05` and `g13` which need 64 and 55 generations respectively. For

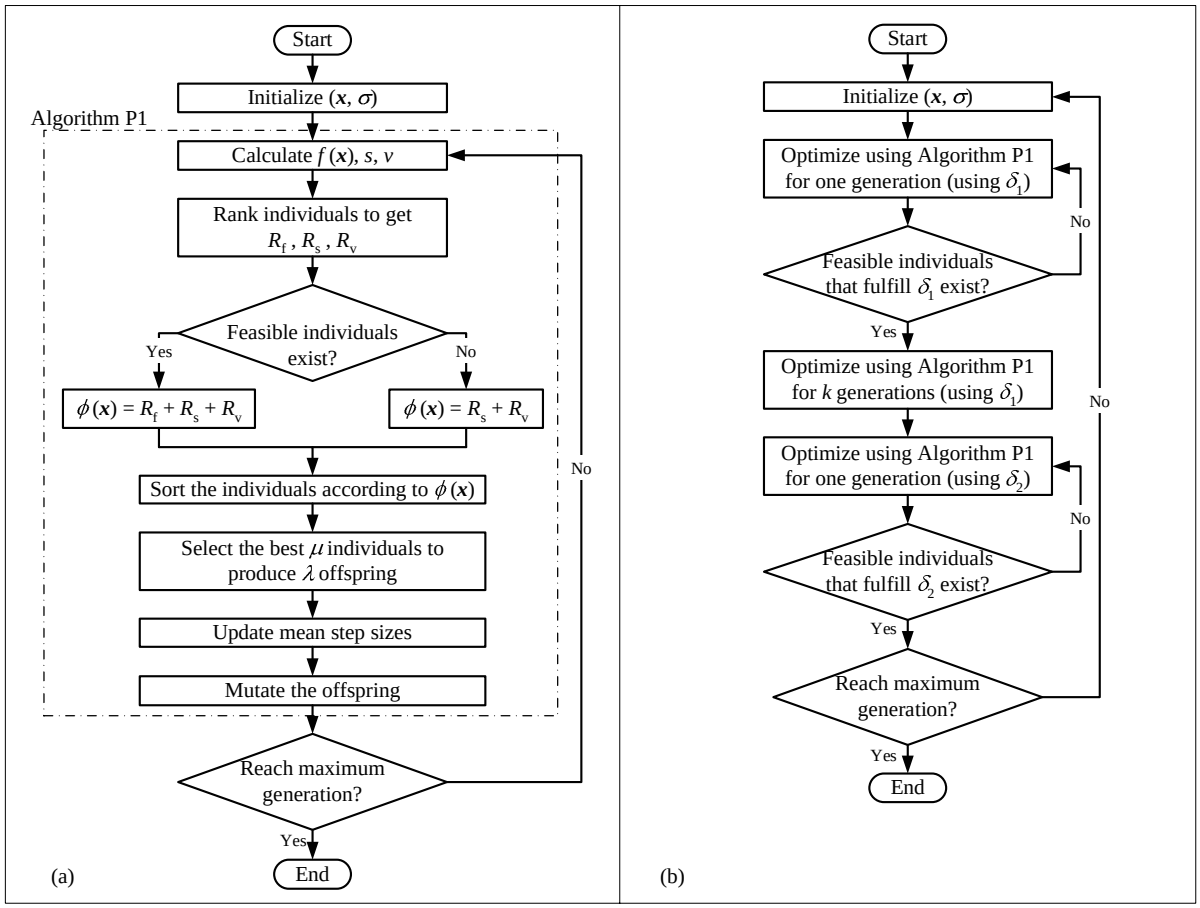

Figure 1: General flow diagram of the (μ, λ)-ES incorporated with simple addition of ranking method. (a) Core algorithm, (b) for problems with equality constraints

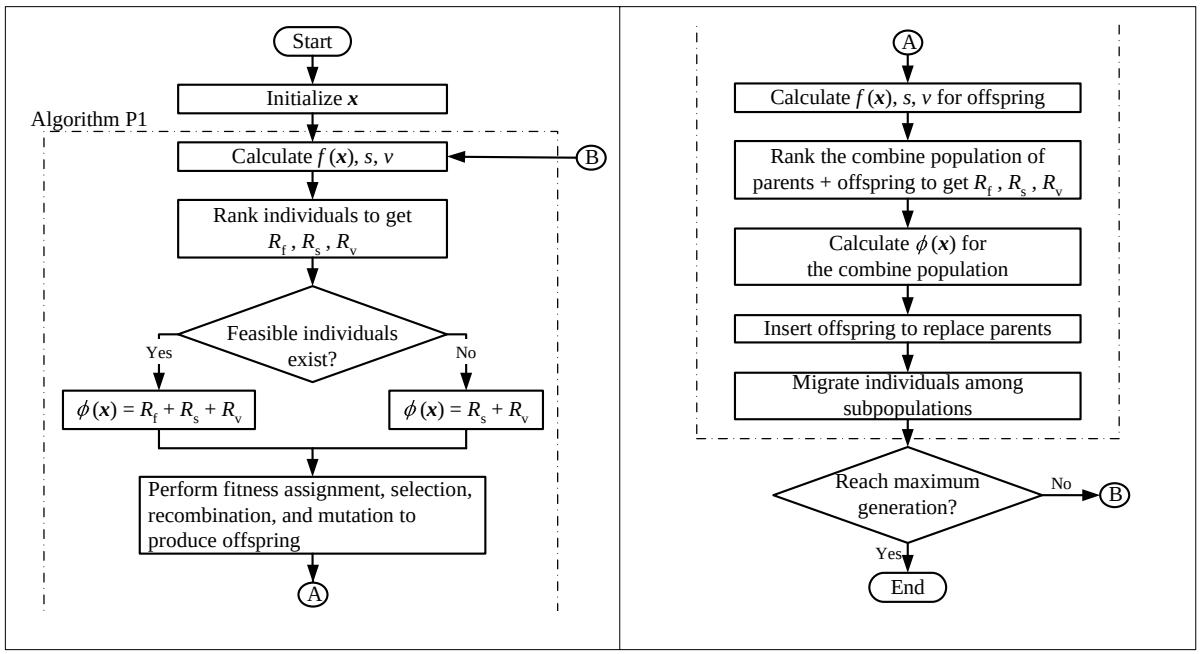

Figure 2: General flow diagram of the GA incorporated with simple addition of ranking method

problems g02, g04, g08, g09, and g12, feasible individuals were found in the randomly initialized population without any optimization. This is consistent with the study from [7] which shows that these five problems have a ratio of feasible region to that of entire search space which is greater than 0.5% ($1/200 \times 100 = 0.5\%$, assuming one feasible individual is found in the population).

The second test is to check the performance of the proposed (μ, λ)-ES algorithm in Fig. 1(a) for all of the problems. Table 3 shows the corresponding experimental results and Table 4 shows a comparison of results to the stochastic ranking method of Runarsson and Yao [9]. Comparison to others results (e.g., [5] and [7]) are not made due to space limitation. Table 3 is separated into two parts: the upper part is for problems with inequality constraints only, the lower part is for problems with equality constraints. G_{m} is the median of generation number to find the best solution in a total of 100 independent runs, while N_{s} is the number of runs where feasible solutions have been found.

For the problems with inequality constraints only, the proposed algorithm was able to find the optimal solutions for g01, g04, g06, g08, g12, and a very close to optimal solutions for g02, g07, and g09. From Table 4, it can be seen that the performance of the proposed strategy was similar to that of stochastic ranking method for g01, g08, and g12. For problems g06 and g10, the mean and worst results obtained here were even better than the stochastic ranking method. For g02, g04, g07, g09, the mean and worst results were only slightly worse than the stochastic ranking method.

However, when dealing with equality constraints, the proposed strategy in Fig. 1(a) could only find the optimal solution for g03, but not in the problems of g05, g11, and g13. For these cases, only a very slight reduction in the objective function value was observed after the first feasible individual was found. The optimization became stagnant in the local minimum and causing the mean and worst results deviated far away from that of [9]. The observation here then induced the idea of adding a simple diversity mechanism and a tolerance value adjustment scheme into the algorithm, as illustrated in Fig. 1(b).

Before continuing, we show the effects of varying the parameters B and k on the performance of the proposed algorithm. Firstly, we varied the B value from 0.001 to 0.3 by fixing the k value as 40. Secondly, we varied the k value from 10 to 50 by fixing the B value as 0.05. The corresponding results are shown in Tables 5 and 6 respectively.

From Table 5, it can be seen that for B values from 0.001 to 0.2, the results for both g05 and g11 were comparable to that of [9] in terms of the best, mean, and worst results. Only when $B = 0.3$, the results were worse than that of [9]. For g13, the results of the proposed strategy were much more better than that of [9] for B values from 0.01 to 0.3. The worst results found in here were by far the best results obtained when compared to others like [5], [7], and [9]. Although Runarsson and Yao [9] is able to obtain 0.056224 as the worst value when P_f in their algorithm is set to 0.475, this result is very sensitive to the setting of P_f. A slight change of P_f from 0.475 to 0.45 can shift the worst value from 0.056224 to 0.216915. The worst values obtained here were consistent for a range of B values from 0.01 to 0.3, indicating that the algorithm is more robust in terms of parameter tuning. Besides, the results obtained here were based on a total of 100 independent runs, instead of 30 as in previous publications, indicating a higher reliability in the results. Problem g13 only failed when B was set to 0.001.

From the study, it was found that the algorithm did not perform well when the B value is either too small (0.001) or too large (0.3). This is because when we set B as 0.001 (0.1% of the $\max(|h_j(\vec{x})|)$), δ_{1j} in (4) becomes too small and this results in restricted size on the search region areas. As a result, the search of solutions that are close to the target region that contain the global minimum becomes more difficult. When the B value increases (0.01 to 0.2), the search region areas become larger and this increases the chances of the algorithm to find solutions near to the target region. However, when we set the B value to 0.3 (30% of $\max(|h_j(\vec{x})|)$), the search region areas become too large and the algorithm may lead us to other feasible regions (containing local minima) which are located far away from the desired target region. From this study, the suitable range for B values was found to be from 0.01 (1%) to 0.2 (20%). It has to be noted that some of the best solutions found were even better than the optimal values, which is due to the approximation of the equality constraints into inequality constraints using (1).

On the other hand, Table 6 shows that when k was varied from 10 to 50 generations, both g05 and g11 gave comparable results to that of [9] in terms of the best, mean, and worst values. However, g13 gave good results only when using k values from 30 to 50. When the k value is large (30 to 50), we allow the algorithm to run for more generations using δ_1 as the tolerance value. By doing so, the solutions found in this stage will have smaller f values when compared to the cases of using small k values (≤ 20). Since the search regions are normally very close to the feasible regions, when we continue the optimization using δ_2 as the tolerance value, we only need to vary slightly on the variables $\vec{x}$ in order to bring them into the feasible regions. As we have only varied slightly on the variables $\vec{x}$, the corresponding f value is also only varied slightly at the same time. In other words, a smaller f value obtained in the search region (δ_1 region) may subsequently lead to a solution which has a smaller f value in the δ_2 region. As a result, larger k value is preferred as this will give us solutions with smaller f values in the δ_1 region. The final settings chosen here were $B = 0.05$ and $k = 40$. A summary of the results is given in Table 7.

The optimization results using GA are shown in Table 8. For g05, g11, and g13, the simple diversity mechanism and tolerance value adjustment scheme in Fig. 1(b) is incorporated, with $B = 0.05$ and $k = 40$. In general, the results were comparable to that of ES-based results. Noticeable differences included a slightly better performance in GA for g04, g06, g10, and a slightly worse performance for g07, g05, g11, and g13. In the case of g11, there were 2 runs which gave solutions of 1.000. Since GA involves much more parameter tuning than that of ES, the parameter settings chosen here may not be the optimal values for the cases here. It has to be noted that there was no attempt in this study to perform fine tuning on the optimization parameters for ES and GA, nor to study the effects of parameter variations on the performance of the proposed algorithm.

Another important finding from this study was that the concept of ranking addition worked equally well even if we neglected the information from the number of constraints violated, R_{v}. We have retested both of the ES and GA by replacing (3) with $\phi(\vec{x}) = R_{\mathrm{f}} + R_{\mathrm{s}}$ and the results are shown

Table 2: Characteristics of the thirteen benchmark problems and feasibility study

Problem	n	Type	LI	NI	LE	NE	$\phi(\vec{x}) = R_s + R_v$			$\phi(\vec{x}) = R_s$		
							N_s	G_{fea}	Std	N_s	G_{fea}	Std
g01	13	quadratic	9	0	0	0	100	13	3	100	12	3
g02	20	nonlinear	1	1	0	0	100	1	0	100	1	0
g03	10	nonlinear	0	0	0	1	100	17	12	100	19	11
g04	5	quadratic	0	6	0	0	100	1	0	100	1	0
g05	4	nonlinear	2	0	0	3	100	64	3	93	63	3
g06	2	nonlinear	0	2	0	0	100	7	2	100	7	2
g07	10	quadratic	3	5	0	0	100	12	2	100	12	2
g08	2	nonlinear	0	2	0	0	100	1	1	100	1	1
g09	7	nonlinear	0	4	0	0	100	1	1	100	1	1
g10	8	linear	3	3	0	0	100	23	5	100	17	4
g11	2	quadratic	0	0	0	1	100	12	4	100	12	5
g12	3	quadratic	0	9^3	0	0	100	1	0	100	1	0
g13	5	nonlinear	0	0	1	2	100	55	4	100	54	4

Table 3: Experimental results using (μ, λ)-ES with simple addition of ranking method (Number in boldface means optimal solution is found)

Prob.	Optimal	Best	Median	Mean	St. Dev.	Worst	G_m	N_s
Problems with inequality constraints only								
g01	-15.000	**-15.000**	**-15.000**	**-15.000**	0.0E+00	**-15.000**	691	100
g02	-0.803619	-0.803602	-0.780828	-0.777351	2.0E-02	-0.712177	1225	100
g04	-30665.539	**-30665.539**	**-30665.539**	-30665.228	9.9E-01	-30659.007	679	100
g06	-6961.814	**-6961.814**	-6952.210	-6896.354	9.5E+01	-6566.977	34	100
g07	24.306	24.307	24.384	24.417	1.2E-01	25.004	533	100
g08	-0.095825	**-0.095825**	**-0.095825**	**-0.095825**	3.0E-17	**-0.095825**	383	100
g09	680.630	680.632	680.649	680.663	3.9E-02	680.863	387	100
g10	7049.25	7063.312	7289.269	7365.964	2.5E+02	8220.442	678	100
g12	-1.000000	**-1.000000**	**-1.000000**	**-1.000000**	0.0E+00	**-1.000000**	63	100
Problems with equality constraints								
g03	-1.000	**-1.000**	**-1.000**	**-1.000**	2.5E-05	**-1.000**	1682	100
g05	5126.498	5126.499	5433.689	5473.997	2.7E+02	6080.091	247	100
g11	0.750	**0.750**	0.826	0.827	6.0E-02	0.957	493	100
g13	0.053950	0.443019	0.997364	0.973519	8.5E-02	0.998250	1750	100

Table 4: Comparison of results between the proposed methodology in ES (A) and the stochastic ranking of [9] (SR) (Number in boldface means better or similar results to that of SR)

Prob.	Optimal	Best Result		Mean Result		Worst Result	
		A	SR	A	SR	A	SR
g01	-15.000	**-15.000**	-15.000	**-15.000**	-15.000	**-15.000**	-15.000
g02	-0.803619	**-0.803602**	-0.803515	-0.777351	-0.781975	-0.712177	-0.726288
g04	-30665.539	**-30665.539**	-30665.539	-30665.228	-30665.539	-30659.007	-30665.539
g06	-6961.814	**-6961.814**	-6961.814	**-6896.354**	-6875.940	**-6566.977**	-6350.262
g07	24.306	**24.307**	24.307	24.417	24.374	25.004	24.642
g08	-0.095825	**-0.095825**	-0.095825	**-0.095825**	-0.095825	**-0.095825**	-0.095825
g09	680.630	680.632	680.630	680.663	680.656	680.863	680.763
g10	7049.25	7063.312	7054.316	**7365.964**	7559.192	**8220.442**	8835.655
g12	-1.000000	**-1.000000**	-1.000000	**-1.000000**	-1.000000	**-1.000000**	-1.000000
g03	-1.000	**-1.000**	-1.000	**-1.000**	-1.000	**-1.000**	-1.000
g05	5126.498	5126.499	5126.497	5473.997	5128.881	6080.091	5142.472
g11	0.750	**0.750**	0.750	0.827	0.750	0.957	0.750
g13	0.053950	0.443019	0.053957	0.973519	0.067543	0.998250	0.216915

Table 5: Effects of varying B on the optimization results, $k = 40$ (Number in boldface means better or similar results to that of SR [9])

Prob.(Optimal)	$B \rightarrow$	0.001	0.01	0.05	0.1	0.2	0.3	SR
g05	Best	5126.498	**5126.497**	5126.498	5126.498	5126.498	5126.498	5126.497
(5126.498)	Mean	**5126.898**	**5126.550**	**5127.179**	5130.083	5129.118	5235.356	5128.881
	Worst	**5135.019**	**5126.967**	**5142.413**	5166.521	5156.092	6100.280	5142.472
	N_s	100	100	95	91	95	86	-
g11	Best	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**	0.759	0.750
(0.750)	Mean	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**	0.760	0.750
	Worst	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**	0.760	0.750
	N_s	100	100	100	100	100	100	-
g13	Best	0.129034	**0.053953**	**0.053952**	**0.053950**	0.053976	0.053993	0.053957
(0.053950)	Mean	0.350095	**0.055216**	**0.054419**	**0.054639**	**0.054892**	**0.055850**	0.067543
	Worst	0.443812	**0.064755**	**0.055600**	**0.056572**	**0.060031**	**0.064886**	0.216915
	N_s	100	100	100	100	100	100	-

Table 6: Effects of varying k on the optimization results, $B = 0.05$ (Number in boldface means better or similar results to that of SR [9])

Prob.(Optimal)	$k \rightarrow$	10	20	30	40	50	SR
g05	Best	5126.498	**5126.497**	5126.498	5126.498	**5126.497**	5126.497
(5126.498)	Mean	**5126.957**	**5126.630**	**5127.026**	**5127.179**	**5128.705**	5128.881
	Worst	**5130.593**	**5127.868**	5162.960	**5142.413**	5161.161	5142.472
	N_s	100	100	99	95	92	-
g11	Best	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**	0.750
(0.750)	Mean	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**	0.750
	Worst	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**	0.750
	N_s	100	100	100	100	100	-
g13	Best	0.443224	0.054003	**0.053952**	**0.053952**	**0.053949**	0.053957
(0.053950)	Mean	0.817467	0.080323	**0.054896**	**0.054419**	**0.054466**	0.067543
	Worst	0.997133	**0.183116**	**0.060928**	**0.055600**	**0.056277**	0.216915
	N_s	100	100	100	100	100	-

Table 7: Experimental results using the proposed (μ, λ)-ES algorithm in Fig. 1(b), $B = 0.05$, $k = 40$ (Number in boldface means optimal solution is found)

Prob.	Optimal	Best	Median	Mean	St. Dev.	Worst	G_m	N_s
g05	5126.498	**5126.498**	5126.587	5127.179	2.2E+00	5142.413	821	95
g11	0.750	**0.750**	**0.750**	**0.750**	1.4E-05	**0.750**	844	100
g13	0.053950	0.053952	0.054290	0.054419	3.9E-04	0.055600	841	100

Table 8: Experimental results using GA with simple addition of ranking method (Number in boldface means optimal solution is found)

Prob.	Optimal	Best	Median	Mean	St. Dev.	Worst	G_m	N_s
g01	-15.000	**-15.000**	**-15.000**	**-15.000**	1.7E-05	-14.9999	1578	100
g02	-0.803619	-0.803595	-0.779878	-0.772451	2.1E-02	-0.714454	1734	100
g03	-1.000	**-1.000**	**-1.000**	**-1.000**	2.1E-05	**-1.000**	1730	100
g04	-30665.539	**-30665.539**	-30665.535	-30665.535	2.4E-03	-30665.526	1606	100
g06	-6961.814	-6961.556	-6958.102	-6957.835	2.4E+00	-6951.137	760	100
g07	24.306	24.397	25.051	25.134	5.7E-01	27.444	1722	100
g08	-0.095825	**-0.095825**	**-0.095825**	**-0.095825**	2.1E-14	**-0.095825**	938	100
g09	680.630	680.634	680.659	680.670	3.1E-02	680.802	1708	100
g10	7049.25	7095.17	7245.82	7282.60	1.6E+02	7968.10	1661	100
g12	-1.000000	**-1.000000**	**-1.000000**	**-1.000000**	7.9E-14	**-1.000000**	162	100
g05	5126.498	5126.502	5131.832	5140.032	1.9E+01	5220.956	1126	70
g11	0.750	**0.750**	**0.750**	0.757	3.5E-02	1.000	880	100
g13	0.053950	0.054284	0.066533	0.072600	2.0E-02	0.152089	863	100

Table 9: Experimental results by using $\phi(\vec{x}) = R_f + R_s$ instead of $\phi(\vec{x}) = R_f + R_s + R_v$ in Condition 2 (Number in boldface means optimal solution is found)

Prob.	Optimal	Best Result		Mean Result		Worst Result	
		ES	GA	ES	GA	ES	GA
g01	-15.000	**-15.000**	**-15.000**	-14.988	**-15.000**	-13.828	**-15.000**
g02	-0.803619	-0.803592	-0.803432	-0.780031	-0.774204	-0.710399	-0.708042
g03	-1.000	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**
g04	-30665.539	**-30665.539**	**-30665.539**	-30664.494	-30665.538	-30641.088	-30665.535
g06	-6961.814	**-6961.814**	-6961.638	-6896.409	-6957.912	-6601.170	-6950.474
g07	24.306	24.308	24.336	24.356	24.411	24.616	24.670
g08	-0.095825	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
g09	680.630	680.631	680.633	680.667	680.658	680.867	680.731
g10	7049.25	7050.864	7113.279	7223.263	7259.945	8809.788	7917.746
g12	-1.000000	**-1.000000**	**-1.000000**	**-1.000000**	**-1.000000**	**-1.000000**	**-1.000000**
g05	5126.498	**5126.498**	5126.499	5126.674	5134.886	5137.700	5190.904
g11	0.750	**0.750**	**0.750**	**0.750**	0.756	**0.750**	1.000
g13	0.053950	**0.053948**	0.054386	0.054306	0.061760	0.055947	0.099296

in Table 9 (with no changes in all other settings). It was interesting to note that there were only slight variations in the best and mean values after removing the term R_v. This shows that the information from the constraint violation to some extent, can be represented solely by the sum of squares value.

The above performance tests have successfully illustrated the effectiveness of using (2) and (3) as the new objective functions in finding the global minimum. With the use of ranking, the information from both of the objective function and constraint violation can be converted into numerical values which are in the same order of magnitude. The ranking methodology used here is fast and direct as the information from the constraint violation are represented by the sum of squares value and the number of constraints violated. This is in contrast to the time consuming pareto ranking which needs to take into account the values of each constraint function separately (e.g., in [1] and [8]). The biggest contribution from this work is the idea of using ranking to convert the information from the objective function and constraint violation into values which are in the same order of magnitude, such that direct summation of the ranking terms can then be used to integrate all of the information into one term. By doing so, both of the feasible and infeasible individuals can be retained to the next generation in an effective way. This is in contrast to the stochastic ranking method of [9], in which only one of the information (either the objective function or the constraint violation) is being considered based on a certain probability.

5. CONCLUSIONS

In conclusions, a simple addition of ranking method has been developed for the handling of constraints in EAs. The proposed methodology is able to integrate the information from both of the objective function and constraint violation in an effective way. The computational cost of the proposed approach is comparatively low since pairwise comparison of individuals is not required. Besides, the algorithm does not require any parameter tuning (except those required by the EAs) when dealing with inequality constraints. Although there are two parameters needed for the case of equality constraints, they are not value specific which are sensitive to slight changes in the values. A wide range of values can be used for B and k, causing the results to be robust towards the tuning. Future works included the extension of the proposed approach into multiobjective optimization problems.

6. REFERENCES

[1] A. Angantyr, J. Andersson, and J.-O. Aidanpaa. Constrained optimization based on a multiobjective evolutionary algorithms. In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, volume 3, pages 1560–1567, Piscataway, New Jersey, December 2003. Canberra, Australia, IEEE Service Center.

[2] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, April 1997.

[3] A. J. Chipperfield, P. J. Fleming, and C. M. Fonseca. Genetic algorithm tools for control systems engineering. In *Proceedings of the Adaptive Computing in Engineering Design and Control*, pages 128–133. Plymouth Engineering Design Centre, 1994.

[4] C. A. Coello Coello. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[5] A. Hernández-Aguirre, S. Botello-Rionda, C. A. Coello Coello, G. Lizárraga-Lizárraga, and E. Mezura-Montes. Handling constraints using multiobjective optimization concepts. *International Journal for Numerical Methods in Engineering*, 59(15):1989–2017, April 2004.

[6] R. Hinterding. Constrained parameter optimisation: equality constraints. In *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 1, pages 687–692, Piscataway, New Jersey, May 2001. IEEE Service Center.

[7] E. Mezura-Montes and C. A. Coello Coello. An improved diversity mechanism for solving constrained optimization problems using a multimembered evolution strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2004)*, pages 700–712, Heidelberg, Germany, June 2004. Seattle, WA, Springer Verlag. Lecture Notes in Computer Science Vol. 3102.

[8] T. Ray, T. Kang, and S. K. Chye. An evolutionary algorithm for constrained optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, July 2000. Morgan Kaufmann.

[9] T. P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

[10] T. P. Runarsson and X. Yao. Constrained evolutionary optimization: the penalty function approach. In R. Sarker, M. Mohammadian, and X. Yao, editors, *Evolutionary optimization*, pages 87–113. Kluwer Academic Publishers, 2002. ISBN: 0-7923-7654-4.