Stochastic Ranking for Constrained Evolutionary Optimization

Thomas P. Runarsson and Xin Yao

***Abstract*—Penalty functions are often used in constrained optimization. However, it is very difficult to strike the right balance between objective and penalty functions. This paper introduces a novel approach to balance objective and penalty functions stochastically, i.e., stochastic ranking, and presents a new view on penalty function methods in terms of the dominance of penalty and objective functions. Some of the pitfalls of naive penalty methods are discussed in these terms. The new ranking method is tested using a (μ, λ) evolution strategy on 13 benchmark problems. Our results show that suitable ranking alone (i.e., selection), without the introduction of complicated and specialized variation operators, is capable of improving the search performance significantly.**

***Index Terms*—Constrained optimization, constraint handling, evolution strategy, penalty functions, ranking.**

I. Introduction

THE general nonlinear programming problem (A) can be formulated as solving the *objective function*

$$\text{minimize} \quad f(\boldsymbol{x}), \qquad \boldsymbol{x} = (x_1, \cdots, x_n) \in \mathcal{R}^n \tag{1}$$

where $\boldsymbol{x} \in \mathcal{S} \cap \mathcal{F}$, $\mathcal{S} \subseteq \mathcal{R}^n$ defines the *search space* which is an n-dimensional space bounded by the *parametric constraints*

$$\underline{x}_i \leq x_i \leq \overline{x}_i, \qquad i \in \{1, \cdots, n\} \tag{2}$$

and the *feasible region* $\mathcal{F}$ is defined by

$$\mathcal{F} = \{\boldsymbol{x} \in \mathcal{R}^n | g_j(\boldsymbol{x}) \leq 0 \qquad \forall j \in \{1, \cdots, m\}\} \tag{3}$$

where $g_j(\boldsymbol{x}), j \in \{1, \cdots, m\}$ are *constraints*.

One common approach to deal with constrained optimization problems is to introduce a penalty term into the objective function to penalize constraint violations [5]. The introduction of the penalty term enables us to transform a constrained optimization problem (A) into an unconstrained one (A'), such as the one given by (4):

$$\psi(\boldsymbol{x}) = f(\boldsymbol{x}) + r_g \phi(g_j(\boldsymbol{x}); \qquad j = 1, \cdots, m) \tag{4}$$

where $\phi \geq 0$ is a real-valued function which imposes a "penalty" controlled by a sequence of *penalty coefficients* $\{r_g\}_0^G$, where g is the generation counter. The general form of function ϕ includes both the generation counter g (for dynamic penalty) and the population (for adaptive penalty). In the current notation, this is reflected in the penalty coefficient r_g. The function ψ will also be referred to as the *fitness function* in this paper.[1] This transformation, i.e., (4), has been used widely in evolutionary constrained optimization [13], [21]. In particular, the following quadratic loss function [5], whose decrease in value represents an approach to the feasible region, has often been used as the *penalty function* [16], [12]:

$$\phi((g_j(\boldsymbol{x}); \qquad j = 1, \cdots, m) = \sum_{j=1}^{m} \max\{0, g_j(\boldsymbol{x})\}^2. \tag{5}$$

We will also use this function here, although our proposed constraint-handling technique is equally applicable to any other forms of penalty functions.

The penalty function method may work quite well for some problems; however, deciding an optimal (or near-optimal) value for r_g turns out to be a difficult optimization problem itself! If r_g is too small, an infeasible solution may not be penalized enough. Hence, an infeasible solution may be evolved by an evolutionary algorithm. If r_g is too large, a feasible solution is very likely to be found, but could be of very poor quality. A large r_g discourages the exploration of infeasible regions, even in the early stages of evolution. This is particularly inefficient for problems where feasible regions in the whole search space are disjoint. In this case, it may be difficult for an evolutionary algorithm to move from one feasible region to another unless they are very close to each other. Reasonable exploration of infeasible regions may act as bridges connecting two or more different feasible regions. The critical issue here is how much exploration of infeasible regions (i.e., how large r_g is) should be considered as reasonable. The answer to this questions is problem dependent. Even for the same problem, different stages of evolutionary search may require different r_g values.

There has been some work on the dynamic setting of r_g values in evolutionary constrained optimization [12], [13], [16]. Such work usually relies on a predefined monotonically nondecreasing sequence of r_g values. This approach worked well for some simple problems, but failed for more difficult ones because the optimal setting of r_g values is problem dependent [19]. A fixed and predefined sequence cannot treat a variety of different problems satisfactorily. A trial-and-error process has to be used in this situation in order to find a proper function for r_g, as is done in [12], [13]. An adaptive approach, where r_g values are adjusted dynamically and automatically by an evolutionary algorithm itself, appears to be most promising in tackling different constrained optimization problems. For example, population information can be used to adjust r_g values adaptively [22]. Different problems lead to different populations in evolutionary search, and thus lead to different r_g values. The advantage of such an adaptive approach is that it can be applied to problems where little prior knowledge is

Manuscript received June 5, 1999; revised November 22, 1999 and April 28, 2000.

T. P. Runarsson is with the Department of Mechanical Engineering, University of Iceland, 107 Reykjavik, Iceland (e-mail: tpr@verk.hi.is).

X. Yao is with the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

Publisher Item Identifier S 1089-778X(00)07038-7.

[1]We are minimizing, rather than maximizing, the fitness function in this paper.

available because we do not need to find a predefined r_g value, or a sequence of r_g values, that is "optimal" for this problem.

According to (4), different r_g values define different fitness functions. A fit individual under one fitness function may not be fit under a different fitness function. Finding a near-optimal r_g adaptively is equivalent to ranking individuals in a population adaptively. Hence, the issue becomes how to rank individuals according to their objective and penalty values. Rank-based selection will be used here. We propose a novel method for ranking individuals without specifying an r_g value. Experimental studies test the effectiveness and efficiency of our method, which can be regarded as an exterior penalty approach.

One approach to avoid setting a hard-to-set parameter r_g is to treat constrained optimization as multiobjective optimization where constraints are regarded as an additional objective function [23], [2]. However, multiobjective optimization does not appear to be any easier than constrained optimization since one has to balance different objectives in optimization.

The rest of this paper is organized as follows. Section II discusses the relationship between r_g and ranking in more details. The concept of dominance is introduced, which is somewhat similar to, but not the same as an early work [21]. The analysis of penalty methods from the point of view of balancing dominance between the objective and penalty functions has revealed what penalty methods are trying to do, and has led to the development of our new constraint-handling technique—stochastic ranking, which balances such dominance directly and explicitly in order to improve the effectiveness and efficiency of constrained algorithms. The relationship between our new technique and previous techniques is also analyzed. Section III describes implementation details of our evolutionary algorithm for constrained optimization, and presents the experimental results on 13 benchmark problems. Comparisons with other constrained optimization algorithms are also included in this section. Finally, Section IV concludes with a brief summary of the paper and a few remarks.

II. Constraint Handling by Stochastic Ranking

A. Penalty Method

For a given penalty coefficient $r_g > 0$, let the ranking of λ individuals be

$$\psi(\boldsymbol{x}_1) \leq \psi(\boldsymbol{x}_2) \leq \cdots \leq \psi(\boldsymbol{x}_\lambda) \tag{6}$$

where ψ is the transformation function given by (4). Let us examine the adjacent pair i and $i+1$ in the ranked order

$$f_i + r_g\phi_i \leq f_{i+1} + r_g\phi_{i+1}, \qquad i \in \{1, \cdots, \lambda - 1\} \tag{7}$$

where the notation $f_i = f(\boldsymbol{x}_i)$ and $\phi_i = \phi(g_j(\boldsymbol{x}_i), j = 1, \cdots, m))$ are used for convenience. We now introduce a parameter $\check{r}_i$, which will be referred to as the *critical penalty coefficient* for the adjacent pair i and $i+1$

$$\check{r}_i = (f_{i+1} - f_i)/(\phi_i - \phi_{i+1}), \qquad \text{for } \phi_i \neq \phi_{i+1}. \tag{8}$$

For the given choice of $r_g \geq 0$, there are three different cases which may give rise to the inequality (7).

1) $f_i \leq f_{i+1}$ and $\phi_i \geq \phi_{i+1}$: The comparison is said to be *dominated by the objective function* and $0 < r_g \leq \check{r}_i$ because the objective function f plays the dominant role in determining the inequality. When individuals are feasible, $\phi_i = \phi_{i+1} = 0$ and $\check{r}_i \to \infty$.
2) $f_i \geq f_{i+1}$ and $\phi_i < \phi_{i+1}$: The comparison is said to be *dominated by the penalty function* and $0 < \check{r}_i < r_g$ because the penalty function ϕ plays the dominant role in determining the inequality.
3) $f_i < f_{i+1}$ and $\phi_i < \phi_{i+1}$: The comparison is said to be *nondominated* and $\check{r}_i < 0$. Neither the objective nor the penalty function can determine the inequality by itself.

When comparing nondominant and feasible individuals, the value of r_g has no impact on the inequality (7). In other words, it does not change the order of ranking of the two individuals. However, the value of r_g is critical in the first two cases as $\check{r}_i$ is the flipping point that will determine whether the comparison is objective or penalty function dominated. For example, if we increase r_g to a value greater than $\check{r}_i$ in the first case, individual $i+1$ would change from a fitter individual into a less-fit one. For the entire population, the chosen value of r_g used for comparisons will determine the fraction of individuals dominated by the objective and penalty functions.

Not all possible r_g values can influence the ranking of individuals. They have to be within a certain range, i.e., $\underline{r}_g < r_g < \overline{r}_g$, to influence the ranking, where the lower bound $\underline{r}_g$ is the minimum critical penalty coefficient computed from adjacent individuals ranked only according to the objective function, and the upper bound $\overline{r}_g$ is the maximum critical penalty coefficient computed from adjacent individuals ranked only according to the penalty function. In general, there are three different categories of r_g values.

1) $r_g < \underline{r}_g$: *All* comparisons are based only on the fitness function. r_g is too small to influence the ranking of individuals. We will call this *underpenalization*.
2) $r_g > \overline{r}_g$: *All* comparisons are based only on the penalty function. r_g is so large that the impact of the objective function can be ignored. We will call this *overpenalization*.
3) $\underline{r}_g < r_g < \overline{r}_g$: *All* comparisons are based on a combination of objective and penalty functions.

All penalty methods can be classified into one of the above three categories. Some methods may fall into different categories during different stages in search. It is important to understand the difference among these three categories because they indicate which function (combination of functions) is driving the search process and how search progresses. For example, most dynamic methods start with a low r_g value (i.e., $r_g < \underline{r}_g$) in order to find a good region which may contain both feasible and infeasible individuals. Toward the end of the search, a high r_g value (i.e., $r_g > \overline{r}_g$) is often used in order to locate a good feasible individual. Such a dynamic method would work well for problems for which the unconstrained global optimum is close to its constrained global optimum. It is unlikely to work well for problems for which the constrained global optimum is

far away from its unconstrained one because the initial low r_g value would drive the search toward the unconstrained global optimum, and thus further away from the constrained one.

The traditional constraint-handling technique used in evolution strategies falls roughly into the category of overpenalization since all infeasible individuals are regarded worse than feasible ones [20], [4], [11]. In fact, canonical evolution strategies (ES) allow only feasible individuals in the initial population. To perform constrained optimization, an ES may be used to find a feasible initial population by minimizing the penalty function [20, p. 115]. Once a feasible population is found, the ES algorithm will then minimize the objective function, and reject all infeasible solutions generated.

It has been widely recognized that neither under- nor overpenalization is a good constraint-handling technique, and there should be a balance between preserving feasible individuals and rejecting infeasible ones [7]. In other words, ranking should be dominated by a combination of objective and penalty functions, and so the penalty coefficient r_g should be within the bounds $\underline{r}_g < r_g < \overline{r}_g$. It is worth noting that the two bounds are not fixed. They are problem dependent, and may change from generation to generation as they are also determined by the current population.

A simple way to measure the balance of dominance of objective and penalty functions is to count how many comparisons of adjacent pairs are dominated by the objective and penalty function, respectively. Such a number of comparisons can be computed for any given r_g by counting the number of critical penalty coefficients given by (8) which are greater than r_g. If we have a predetermined preference for the number of adjacent comparisons that should be dominated by the penalty function, then a corresponding penalty coefficient could be found.

It is clear from the analysis in this section that all a penalty method tries to do is to obtain the right balance between objective and penalty functions so that the search moves toward the optimum in the feasible space, not just toward the optimum in the combined feasible and infeasible space. One way to achieve such balancing effectively and efficiently is to adjust such balance directly and explicitly. This is what stochastic ranking, described in the next section, does.

B. Stochastic Ranking

Since the optimal r_g is hard to determine, a different approach is used here to balance the dominance of the objective and penalty functions. We introduce a probability P_f of using only the objective function for comparisons in ranking in the infeasible regions of the search space. That is, given any pair of two adjacent individuals, the probability of comparing them (in order to determine which one is fitter) according to the objective function is 1 if both individuals are feasible; otherwise, it is P_f. This appears to be similar to the use of a probability by Surry and Radcliffe [23] in deciding the outcome of competitions between two individuals in tournament selection. Our technique is, however, quite different because we use rank-based selection, and we do not have any extra computational cost for self-adapting P_f. More importantly, the motivation of stochastic ranking comes from the need for balancing objective and penalty functions directly and explicitly in optimization. Surry and Radcliffe's method [23] does not attempt to balance the dominance of penalty and objective functions in a population.

```
I_j = j ∀ j ∈ {1, ..., λ}
for i = 1 to N do
    for j = 1 to λ − 1 do
        sample u ∈ U(0, 1)
        if (φ(I_j) = φ(I_{j+1}) = 0) or (u < P_f) then
          if (f(I_j) > f(I_{j+1})) then
            swap(I_j, I_{j+1})
          fi
        else
          if (φ(I_j) > φ(I_{j+1})) then
            swap(I_j, I_{j+1})
          fi
        fi
    od
    if no swap done break fi
od
```

Fig. 1. Stochastic ranking using a bubble-sort-like procedure where $U(0,1)$ is a uniform random number generator and N is the number of sweeps going through the whole population. When $P_f = 0$, the ranking is an *overpenalization*, and for $P_f = 1$, the ranking is an *underpenalization*. The initial ranking is always generated at random.

Ranking is achieved by a bubble-sort-like procedure[2] in our work. The procedure provides a convenient way of balancing the dominance in a ranked set. In our bubble-sort-like procedure, λ individuals are ranked by comparing adjacent individuals in at least λ sweeps.[3] The procedure is halted when no change in the rank ordering occurs within a complete sweep. Fig. 1 shows the stochastic bubble-sort procedure used to rank individuals in a population.

The probability of an adjacent individual winning a comparison, i.e., holding the higher rank, in the ranking procedure is

$$P_w = P_{fw}P_f + P_{\phi w}(1 - P_f) \tag{9}$$

given that at least one individual is infeasible. P_{fw} is the probability of the individual winning according to the objective function, and $P_{\phi w}$ is the probability of the individual winning according to the penalty function. In the case where adjacent individuals are both feasible, $P_w = P_{fw}$. We would like to examine the probability of winning k more comparisons than losses. Then the total number of wins must be $k' = (N+k)/2$, where N is the total number of comparisons made. The probability of winning k' comparisons out of N is given by the binomial distribution[4]

$$P_w(y = k') = \binom{N}{k'} P_w^{k'}(1 - P_w)^{N-k'}. \tag{10}$$

[2]It can be regarded as the stochastic version of the classic bubble sort.

[3]It would be exactly λ sweeps if the comparisons were not made stochastic.

[4]The standard deviation of the binomial distribution is $\sqrt{NP_w(1-P_w)}$.

TABLE I
AVERAGE PERCENTAGE FEASIBLE INDIVIDUALS IN THE FINAL POPULATION AS A FUNCTION OF P_f ($N = \lambda$) AND TEST FUNCTION PRESENTED IN APPENDIX

fcn\P_f	0.525	0.500	0.475	0.450	0.000
g01	0	0	11	83	82
g02	4	30	50	57	58
g03	0	0	0	2	22
g04	0	17	78	78	80
g05	0	0	78	82	81
g06	0	0	0	53	81
g07	0	1	71	80	79
g08	0	100	100	100	100
g09	0	16	83	83	86
g10	0	0	0	74	73
g11	0	0	85	90	70
g12	100	100	100	100	5
g13	0	0	73	79	44

The probability of winning *at least* k' comparisons is

$$P'_w(y \geq k') = 1 - \sum_{j=0}^{k'-1} \binom{N}{j} P_w^j (1 - P_w)^{N-j}. \qquad (11)$$

Equations (10) and (11) show that the greater the number of comparisons (N), the less influence the initial ranking will have. It is worth noting that the probability P_w is usually different for different individuals in different stages of ranking (sorting). Now, consider a case where P_w is constant during the entire ranking procedure, which is the case when $f_i < f_j$, $\phi_i > \phi_j$; $j \neq i, j = 1, \ldots, \lambda$. Then $P_{fw} = 1$ and $P_{\phi w} = 0$. If we choose $P_f = 1/2$, then $P_w = 1/2$. There will be an equal chance for a comparison to be made based on the objective or penalty function. Since we are only interested in feasible individuals as final solutions, P_f should be less than 1/2 so that there is a bias against infeasible solutions. The strength of the bias can be adjusted easily by adjusting only P_f. When parameter N, the number of sweeps, approaches ∞, then the ranking will be determined by the bias P_f. That is, if $P_f > 1/2$, the ranking is based on the objective function, and when $P_f < 1/2$, the ranking is the overpenalty ranking. Hence, an increase in the number of ranking sweeps is effectively equivalent to changing parameter P_f, i.e., making it smaller or larger. Thus, we can fix $N = \lambda$, and adjust P_f to achieve the best performance. We illustrate these points by optimizing a set of benchmark functions presented in the Appendix using different P_f values. Table I presents the average results over 30 independent runs of our algorithm. The numbers in the Table Indicate the percentage of feasible individuals in the final population. The details about the experiment will be given in the following section. It is quite clear from the table that, as $P_f > 1/2$, finding feasible solutions becomes very difficult unless the unconstrained optimum happens to be the same as the constrained optimum, as is the case for problem `g12`.

III. EXPERIMENTAL STUDIES

A. Evolution Strategy

The evolutionary optimization algorithm described in this section is based on ES [20]. One reason for choosing ES is that it does not introduce any specialized constraint-handling variation operators. We would like to show that specialized and complex variation operators for constrained optimization problems are unnecessary, although they may be quite useful for particular types of problems (see, for example, [17]). A simple extension to the ES, i.e., the use of the stochastic ranking scheme proposed in the previous section, can achieve significantly better results than other more complicated techniques. The constraint-handling technique based on the stochastic ranking scheme can be used in any evolutionary algorithm, not just ES.

In the (μ, λ)-ES algorithm, the individual i is a set of real-valued vectors $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i), \forall i \in \{1, \cdots, \lambda\}$. The initial population of $\boldsymbol{x}$ is generated according to a uniform n-dimensional probability distribution over the search space $\mathcal{S}$. Let δx be an approximate measure of the expected distance to the global optimum, then the initial setting for the "mean step sizes" should be [20, p. 117]

$$\sigma_{i,j}^{(0)} = \delta x_j / \sqrt{n} \approx (\overline{x}_j - \underline{x}_j)/\sqrt{n}, \quad i \in \{1, \ldots, \lambda\}, j \in \{1, \ldots, n\} \qquad (12)$$

where $\sigma_{i,j}$ denotes the jth component of the vector $\boldsymbol{\sigma}_i$. We use these initial values as upper bounds on $\boldsymbol{\sigma}$.

Following the stochastic ranking scheme given previously, the evaluated objective $f(\boldsymbol{x})$ and penalty function $\phi(g_k(\boldsymbol{x}); k = 1, \ldots, m)$ for each individual $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i), \forall i \in \{1, \ldots, \lambda\}$ are used to rank the individuals in a population, and the best (highest ranked) μ individuals out of λ are selected for the next generation. The truncation level is set at $\mu/\lambda \approx 1/7$ [1, p. 79].

Variation of strategy parameters is performed before the modification of objective variables. We generate λ new strategy parameters from μ old ones so that we can use the λ new strategy parameters in generating λ offspring later. The "mean step sizes" are updated according to the log-normal update rule [20]: $i = 1, \ldots, \mu$, $h = 1, \ldots, \lambda$, and $j = 1, \ldots, n$

$$\sigma_{h,j}^{(g+1)} = \hat{\sigma}_{h,j}^{(g)} \exp(\tau' N(0,1) + \tau N_j(0,1))) \qquad (13)$$

where $N(0,1)$ is a normally distributed one-dimensional random variable with an expectation of 0 and variance 1. The subscript j in $N_j(0,1)$ indicates that the random number is generated anew for each value of j. The "learning rates" τ and τ' are set equal to $\varphi^*/\sqrt{2\sqrt{n}}$ and $\varphi^*/\sqrt{2n}$, respectively, where φ^* is the expected rate of convergence [20, p. 144] and is set to 1 [1, p. 72]. Recombination is performed on the self-adaptive parameters before applying the update rule given by (13). In particular, global intermediate recombination (the average) between two parents [20, p. 148] is implemented as

$$\hat{\sigma}_{h,j}^{(g)} = (\sigma_{i,j}^{(g)} + \sigma_{k_j,j}^{(g)})/2, \qquad k_j \in \{1, \ldots, \mu\} \qquad (14)$$

where k_j is an index generated at random and anew for each j.

TABLE II
EXPERIMENTAL RESULTS ON 13 BENCHMARK FUNCTIONS USING ES WITH STOCHASTIC RANKING ($P_f = 0.45$); 30 INDEPENDENT RUNS WERE CARRIED OUT

fcn	optimal	best	median	mean	st. dev.	worst	G_m
g01	−15.000	−15.000	−15.000	−15.000	0.0E+00	−15.000	741
g02	−0.803619	−0.803515	−0.785800	−0.781975	2.0E−02	−0.726288	1086
g03	−1.000	−1.000	−1.000	−1.000	1.9E−04	−1.000	1146
g04	−30665.539	−30665.539	−30665.539	−30665.539	2.0E−05	−30665.539	441
g05	5126.498	5126.497	5127.372	5128.881	3.5E+00	5142.472	258
g06	−6961.814	−6961.814	−6961.814	−6875.940	1.6E+02	−6350.262	590
g07	24.306	24.307	24.357	24.374	6.6E−02	24.642	715
g08	−0.095825	−0.095825	−0.095825	−0.095825	2.6E−17	−0.095825	381
g09	680.630	680.630	680.641	680.656	3.4E−02	680.763	557
g10	7049.331	7054.316	7372.613	7559.192	5.3E+02	8835.655	642
g11	0.750	0.750	0.750	0.750	8.0E−05	0.750	57
g12	−1.000000	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	82
g13	0.053950	0.053957	0.057006	0.067543	3.1E−02	0.216915	349

Having varied the strategy parameters, each individual $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$, $\forall i \in \{1, \cdots, \mu\}$ creates λ/μ offspring on average, so that a total of λ offspring are generated

$$x_{h,j}^{(g+1)} = x_{i,j}^{(g)} + \sigma_{h,j}^{(g+1)} N_j(0,1). \tag{15}$$

Recombination is not used in the variation of objective variables. When an offspring is generated outside the parametric bounds defined by the problem, the mutation (variation) of the objective variable will be retried until the variable is within its bounds. In order to save computation time, the mutation is retried only ten times and then ignored, leaving the object variable in its original state within the parameter bounds.

B. Experimental Results and Discussions

Thirteen benchmark functions were used. The first 12 were taken from [14], and the 13th from [15]. The details, including the original sources, of these functions are listed in the Appendix. Problems g02, g03, g08, and g12 are maximization problems. They were transformed into minimization problems using $-f(\boldsymbol{x})$. For each of the benchmark problems, 30 independent runs were performed using a (30, 200)-ES. All experiments were performed in MATLAB. The source code may be obtained from the authors upon request. All runs were terminated after $G = 1750$ generations, except for g12, which was run for 175 generations. Problem g12 is the harder version studied in [14], where the feasible region of the search space consists of 9^3 disjointed spheres with a radius of 0.25.

Table II summarizes the experimental results we obtained using $P_f = 0.45$. The median number of generations for finding the best solution in each run is indicated by G_m in the Table. The table also shows the known "optimal" solution for each problem and statistics for the 30 independent runs. These include the best objective value found, median, mean, standard deviation, and worst found. The statistics are based on feasible solutions only. All equality constraints have been converted into inequality constraints, $|h(\boldsymbol{x})| - \delta \leq 0$, using the degree of violation $\delta = 0.0001$. As a result of this approximation, some results might be better than the optimum. However, the tolerated violation is more stringent than others [15] where $\delta = 0.001$ was used.

In comparison with the latest results in the literature [14], the results in Table II are significantly better for all but one problem. While $70 \times 20\,000$ function evaluations were used for each problem and only 20 runs were carried out in [14] (for Experiment 2 in [14], which gave better results than Experiment 1), we have used a maximum of only 200×1750 function evaluations for each problem, and carried out 30 independent runs for all problems.

For problems g01, g03, g04, g08, g11, and g12, our algorithm has consistently found the optimal solution for all 30 runs, while the algorithm in [14] did not find any for problems g01, g03, g04, and g12 (the more difficult version), and found the optimal solution only in some out of 20 runs for problem g08. The average result in [14] for g08 was −0.0891568 when the optimal solution was −0.095825.

For problem g02, the algorithm given by [14] was more consistent and performed better on average, but worse in terms of the best result. Its average result was −0.79671 over 20 runs, while ours was −0.781975 over 30 runs.[5] However, our algorithm was capable of finding better solutions. The best solution found by our algorithm was −0.803515, while the best in [14] was −0.79953. It is also interesting to note that the median of our results was −0.785800, which was much better than the average. A closer look at our results revealed that six out of 30 runs obtained a solution better than the best offered in [14].

For problem g04, −30664.5 was reported as being "by far the best value reported by any evolutionary system for this test case!" [14]. Our algorithm has now improved this "record" substantially by finding the optimum consistently.[6] Homaifar *et al.*

[5]The minus sign was added to the average result because we transformed the maximization problem into the minimization one.

[6]After this paper had been submitted, Petrowski and Hamida [18] reported another algorithm which could also find the optimum consistently. However, few details about the algorithm, the parameters used, and experimental setup were described in their one-page paper. The optimal solution found by them was only given for one digit after the decimal point. Problems g03, g05, g11, g12, and g13 were not included in their studies.

TABLE III
COMPARISON BETWEEN OUR (INDICATED BY RY) AND KOZIEL AND MICHALEWICZ'S (INDICATED BY KM [14]) ALGORITHMS; FOR PROBLEM g13, THE RESULT WAS TAKEN FROM [15, METHOD 4]; THE TWO VALUES IN THE "MEAN" COLUMN FOR PROBLEM g13 REPRESENT MEDIANS

		Best Result		Mean Result		Worst Result	
fcn	optimal	RY	KM	RY	KM	RY	KM
g01	−15.000	−15.000	−14.7864	−15.000	−14.7082	−15.000	−14.6154
g02	−0.803619	−0.803515	−0.79953	−0.781975	−0.79671	−0.726288	−0.79119
g03	−1.000	−1.000	−0.9997	−1.000	−0.9989	−1.000	−0.9978
g04	−30665.539	−30665.539	−30664.5	−30665.539	−30655.3	−30665.539	−30645.9
g05	5126.498	5126.497	—	5128.881	—	5142.472	—
g06	−6961.814	−6961.814	−6952.1	−6875.940	−6342.6	−6350.262	−5473.9
g07	24.306	24.307	24.620	24.374	24.826	24.642	25.069
g08	−0.095825	−0.095825	−0.0958250	−0.095825	−0.0891568	−0.095825	−0.0291438
g09	680.630	680.630	680.91	680.656	681.16	680.763	683.18
g10	7049.331	7054.316	7147.9	7559.192	8163.6	8835.655	9659.3
g11	0.750	0.750	0.75	0.750	0.75	0.750	0.75
g12	−1.000000	−1.000000	−0.999999857	−1.000000	−0.999134613	−1.000000	−0.991950498
g13	0.053950	0.053957	0.054	0.057006	0.064	0.216915	0.557

[10] found a similar solution to ours for g04 using a genetic algorithm. Unfortunately, that solution violated two constraints. Another similar solution was found by Colville [3] using a mathematical programming technique. However, it is unclear how those two techniques [10], [3] would perform on a larger set of benchmark functions as we used here.

For problem g05, which involves equality constraints, the algorithm given in [14] "did not provide quality results." Hence, no results were given in their paper. Our algorithm has found consistently feasible solutions. Some very good results were obtained. For example, the best result found was 5126.497, and the average was 5128.811. The best result was even better than the optimal solution of 5126.498. This is the consequence of using inequalities to approximate each equality, although we used a very small δ.

For problem g06, our algorithm performed significantly better than the algorithm in [14] in terms of the average as well as best results. Our average result was −6875.940, while theirs was −6342.6. Our algorithm has found the global optimum 20 times out of 30 runs, while their algorithm had never found the optimal solution.[7]

For problems g07, g09, and g10, our algorithm outperformed the algorithm given in [14], again in terms of all three criteria: the average, best, and worst results. Both algorithms performed well, and found the optimal solution for problem g11.

For problem g13, our algorithm outperformed all six constraint-handling methods studied in [15] in terms of the best, median, and worst results. Table III summarizes the comparison between our results and the latest results [14], [23] that we can find in the literature.

In order to evaluate the impact of P_f on the results generated by our algorithm, we have run the same set of experiments many times using $P_f = 0, 0.025, \cdots, 0.525$. As expected, neither small nor large (i.e., >0.5) P_f gave very good results. The best results were obtained when $0.4 < P_f < 0.5$. This indicates that a minor bias toward the dominance of the penalty function encourages the evolution of feasible solutions while still maintaining infeasible regions as potential "bridges" to move among feasible regions in the whole search space.

Tables IV and V give two sets of our experimental results when $P_f = 0$ and $P_f = 0.475$, respectively. $P_f = 0$ is an extreme case where all infeasible individuals were ranked lower than feasible individuals. Among feasible solutions, the ranking was based solely on the objective function. Among infeasible solutions, the ranking was based only on the penalty function. This extreme case is somewhat similar to [4], but not the same because it does not use the worst fitness value of feasible solutions. Although this algorithm did not perform as well as when $P_f = 0.45$ for problems g03, g04, g05, g11, g12, and g13, it performed roughly the same as when $P_f = 0.45$ for other problems. When $P_f = 0.475$, the penalty against the infeasible solution was weakened. Our algorithm could only find a feasible solution 6 times out of 30 runs for problem g10, although it found a feasible solution 100% times for all other problems. In general, the algorithm improved its performance and found best solutions when P_f was changed from 0.45 to 0.475, except for problems g01, g03, and g06. The improvement is especially noticeable for functions g13 and g04.

It is important to emphasize that the performance of any evolutionary algorithm for constrained optimization is determined by the constraint-handling technique used, as well as the evolutionary search algorithm (including parameters). Throughout our study, we have kept our modification to the ES to the minimum, i.e., changing only the selection scheme without introducing any specialized operators. The parameters were also set according to previous recommendations in published books and papers. This, however, does not imply that the search algorithm plays an unimportant role in constrained optimization. To illustrate that the combined effect of a constraint-handling technique

[7]Our algorithm consistently will find the optimum when $P_f = 0.425$; see also the results for $P_f = 0$ in Table IV.

TABLE IV
EXPERIMENTAL RESULTS ON 13 BENCHMARK FUNCTIONS USING ES WITH STOCHASTIC RANKING ($P_f = 0$); 30 INDEPENDENT RUNS WERE CARRIED OUT

fcn	optimal	best	median	mean	st. dev.	worst	G_m
g01	−15.000	−15.000	−15.000	−15.000	0.0E+00	−15.000	697
g02	−0.803619	−0.803578	−0.785253	−0.783049	1.5E−02	−0.750656	1259
g03	−1.000	−0.327	−0.090	−0.105	7.2E−02	−0.014	61
g04	−30665.539	−30665.539	−30665.538	−30664.710	3.8E+00	−30644.897	632
g05	5126.498	5126.945	5225.100	5348.683	2.7E+02	6050.566	213
g06	−6961.814	−6961.814	−6961.814	−6961.814	1.9E−12	−6961.814	946
g07	24.306	24.322	24.367	24.382	5.9E−02	24.598	546
g08	−0.095825	−0.095825	−0.095825	−0.095825	2.7E−17	−0.095825	647
g09	680.630	680.632	680.657	680.671	3.8E−02	680.772	414
g10	7049.331	7117.416	7336.280	7457.597	3.4E+02	8464.816	530
g11	0.750	0.750	0.953	0.937	5.4E−02	0.973	1750
g12	−1.000000	−0.999972	−0.999758	−0.999766	1.4E−04	−0.999488	90
g13	0.053950	0.919042	0.997912	0.993372	1.5E−02	0.998316	1750

TABLE V
EXPERIMENTAL RESULTS ON 13 BENCHMARK FUNCTIONS USING ES WITH STOCHASTIC RANKING ($P_f = 0.475$); 30 INDEPENDENT RUNS WERE CARRIED OUT

fcn	optimal	best	median	mean	st. dev.	worst	G_m
g01	−15.000	−15.000	−5.736	−6.793	3.5E+00	−2.356	951
g02	−0.803619	−0.802760	−0.792272	−0.786084	1.8E−02	−0.731226	1301
g03	−1.000	−0.998	−0.995	−0.995	2.1E−03	−0.990	935
g04	−30665.539	−30665.539	−30665.539	−30665.539	1.1E−11	−30665.539	349
g05	5126.498	5126.518	5127.276	5128.538	3.1E+00	5141.085	448
g06	−6961.814	−6871.345	−6603.846	−6572.309	2.1E+02	−6058.588	14
g07	24.306	24.307	24.317	24.328	2.6E−02	24.392	1568
g08	−0.095825	−0.095825	−0.095825	−0.095825	2.7E−17	−0.095825	463
g09	680.630	680.630	680.634	680.640	1.4E−02	680.676	1449
g10*)	7049.331	7202.108	7343.603	7384.116	1.8E+02	7688.864	1321
g11	0.750	0.750	0.750	0.750	8.7E−06	0.750	118
g12	−1.000000	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	69
g13	0.053950	0.053945	0.054000	0.054179	5.0E−04	0.056224	573

*Based on the six feasible solutions found out of the 30 runs.

TABLE VI
EXPERIMENTAL RESULT ON FUNCTION g10 USING ES WITH STOCHASTIC RANKING AND $\varphi^* = 1/4$

P_f	optimal	best	median	mean	st. dev.	worst	G_m
0.45	7049.331	7049.852	7054.111	7056.163	5.7E+00	7068.633	1733
0.00	7049.331	7049.955	7062.673	7074.044	3.1E+01	7196.647	1745

and a search algorithm can make a big difference, we repeated the experiment on function g10 using $\varphi^* = 1/4$ (instead of 1) for the computation of the learning rates τ and τ'. These results are given in Table VI. A significant improvement was achieved in comparison with the results in Tables II and IV.

An interesting question that arises naturally here is whether or not ES was fully responsible for the good results obtained, e.g., those in Table II, in other words, whether stochastic ranking contributed anything to the good results. To answer this question, an additional set of experiments was carried out using exactly the same ES as used before, but with a different constraint-handling technique—the dynamic penalty method of [12]. The results are summarized in Tables VII and VIII.

Comparing Tables II and VII, it is clear that stochastic ranking performed better than the dynamic penalty method with $r_g = g/2$ [12] according to all four criteria (*best*, *median*, *mean*, and *worst*) for all benchmark functions, except for g02, g09, and g12. The two methods performed the same on problem g12. The dynamic penalty method found a better *best* than stochastic ranking for problem g02, but performed worse than stochastic ranking according to *median*, *mean*, and *worst*. On the other hand, stochastic ranking found a better *best* (i.e., the optimum)

TABLE VII
EXPERIMENTAL RESULTS USING THE DYNAMIC PENALTY METHOD OF [12] WITH $r_g = g/2$; THE SUBSCRIPT IN THE FUNCTION NAME INDICATES THE NUMBER OF FEASIBLE SOLUTIONS FOUND IF IT WAS BETWEEN 1 AND 29 INCLUSIVE; "–" MEANS NO FEASIBLE SOLUTIONS WERE FOUND

fcn	optimal	best	median	mean	st. dev.	worst	G_m
g01	−15.000	−14.990	−14.970	−14.968	1.3E−02	−14.943	122
g02	−0.803619	−0.803597	−0.783042	−0.777241	2.3E−02	−0.710725	1007
g03	−1.000	–	–	–	–	–	–
g04	−30665.539	−30648.710	−30007.572	−30021.805	1.7E+02	−29804.553	5
g05	5126.498	–	–	–	–	–	–
g06	−6961.814	−6897.969	−6534.206	−6502.478	2.3E+02	−5962.775	12
g07	24.306	24.347	24.417	24.479	1.6E−01	24.934	109
g08	−0.095825	−0.095825	−0.095825	−0.095354	1.5E−03	−0.087752	278
g09	680.630	680.632	680.638	680.648	2.7E−02	680.761	109
g10	7049.331	–	–	–	–	–	–
$g11_{(16)}$	0.750	0.750	0.750	0.758	2.5E−02	0.850	14
g12	−1.000000	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	65
$g13_{(25)}$	0.053950	0.281242	0.448114	0.474460	1.1E−01	0.901263	1750

TABLE VIII
EXPERIMENTAL RESULTS USING THE DYNAMIC PENALTY METHOD OF [12] WITH $r_g = (g/2)^2$; "–" MEANS NO FEASIBLE SOLUTIONS WERE FOUND

fcn	optimal	best	median	mean	st. dev.	worst	G_m
g01	−15.000	−15.000	−15.000	−15.000	7.9E−05	−15.000	217
g02	−0.803619	−0.803587	−0.785907	−0.784868	1.5E−02	−0.751624	1235
g03	−1.000	−0.583	−0.045	−0.103	1.4E−01	−0.001	996
g04	−30665.539	−30365.488	−30060.607	−30072.458	1.2E+02	−29871.442	4
g05	5126.498	–	–	–	–	–	–
g06	−6961.814	−6911.247	−6547.354	−6540.012	2.6E+02	−5868.028	13
g07	24.306	24.309	24.375	24.421	2.2E−01	25.534	180
g08	−0.095825	−0.095825	−0.095825	−0.095825	2.8E−17	−0.095825	421
g09	680.630	680.632	680.648	680.659	3.2E−02	680.775	1739
g10	7049.331	–	–	–	–	–	–
g11	0.750	0.750	0.750	0.750	9.1E−06	0.750	61
g12	−1.000000	−1.000000	−0.999818	−0.999838	1.3E−04	−0.999573	68
g13	0.053950	0.514152	0.996674	0.965397	9.4E−02	0.998156	1750

for problem g09, but performed worse than the dynamic penalty method according to *median*, *mean*, and *worst*.

The results in Table VIII with $r_g = (g/2)^2$ improved those in Table VII for most, but not all, problems. Feasible solutions can now be found for problem g03. The results for several problems are now better than those in Table VII. However, none of these improvements has changed the general picture. The results from stochastic ranking are still better than those from the dynamic penalty method [12] with $r_g = (g/2)^2$. In fact, the dynamic penalty method with $r_g = (g/2)^2$ is only better than stochastic ranking for problem g02, but has lost its advantage for problem g09. This is not very surprising because the dynamic penalty method relies on a predefined sequence of r_g, while stochastic ranking is an adaptive method without any predefined sequence.

Predefined sequences are unable to adjust r_g according to different problems and different search stages for a problem. While a predefined sequence may work well for one problem, it may not for a different problem. This is what happened to the dynamic penalty method. On the other hand, an adaptive method like stochastic ranking can adjust the balance between objective and penalty functions automatically for different problems, and during different stages of evolutionary search. However, there is no free lunch in optimization [24]. The price paid by an adaptive method is slightly longer search time for the algorithm to adapt. This can be observed from the last column in Tables II and VII.

IV. CONCLUSION

This paper has proposed a new constraint handling technique—stochastic ranking. The technique does not introduce any specialized variation operators. It does not require *a priori* knowledge about a problem since it does not use any penalty coefficient r_g in a penalty function. Stochastic ranking is motivated by our analysis of penalty methods from the point of view of dominance. The balance between the objective and penalty functions is achieved through a ranking procedure

based on the stochastic bubble-sort algorithm. The introduction of a single probability of P_f enables us to specify conveniently an agreeable bias toward the objective function in ranking individuals. Our experimental results suggest that a value of $0.4 < P_f < 0.5$ would be appropriate for many constrained optimization problems. The new constraint-handling technique was tested on a set of 13 benchmark problems. Experimental results have been presented. The future work of this study includes the application of stochastic ranking to other types of evolutionary algorithms.

Appendix
Test Function Suite

All benchmark functions with the exception of `g13` are described in [14]. They are summarized here for completeness. The original sources of the functions are also cited.

A. `g01`

Minimize [6]

$$f(\boldsymbol{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{16}$$

subject to

$$\begin{aligned}
g_1(\boldsymbol{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0\\
g_2(\boldsymbol{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0\\
g_3(\boldsymbol{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0\\
g_4(\boldsymbol{x}) &= -8x_1 + x_{10} \le 0\\
g_5(\boldsymbol{x}) &= -8x_2 + x_{11} \le 0\\
g_6(\boldsymbol{x}) &= -8x_3 + x_{12} \le 0\\
g_7(\boldsymbol{x}) &= -2x_4 - x_5 + x_{10} \le 0\\
g_8(\boldsymbol{x}) &= -2x_6 - x_7 + x_{11} \le 0\\
g_9(\boldsymbol{x}) &= -2x_8 - x_9 + x_{12} \le 0
\end{aligned} \tag{17}$$

where the bounds are $0 \le x_i \le 1$ $(i = 1, \cdots, 9)$, $0 \le x_i \le 100$ $(i = 10, 11, 12)$ and $0 \le x_{13} \le 1$. The global minimum is at $\boldsymbol{x}^*$ = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1) where six constraints are active (g_1, g_2, g_3, g_7, g_8, and g_9) and $f(\boldsymbol{x}^*) = -15$.

B. `g02`

Maximize [14]

$$f(\boldsymbol{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{18}$$

subject to

$$\begin{aligned}
g_1(\boldsymbol{x}) &= 0.75 - \prod_{i=1}^{n} x_i \le 0\\
g_2(\boldsymbol{x}) &= \sum_{i=1}^{n} x_i - 7.5n \le 0
\end{aligned} \tag{19}$$

where $n = 20$ and $0 \le x_i \le 10$ $(i = 1, \cdots, n)$. The global maximum is unknown; the best we found is $f(\boldsymbol{x}^*) = 0.803619$ (which, to the best of our knowledge, is better than any reported value), constraint g_1 is close to being active ($g_1 = -10^{-8}$).

C. `g03`

Maximize [17]

$$f(\boldsymbol{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i \tag{20}$$

$$h_1(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0 \tag{21}$$

where $n = 10$ and $0 \le x_i \le 1$ $(i = 1, \cdots, n)$. The global maximum is at $x_i^* = 1/\sqrt{n}$ $(i = 1, \cdots, n)$ where $f(\boldsymbol{x}^*) = 1$.

D. `g04`

Minimize [8]

$$\begin{aligned}
f(\boldsymbol{x}) = {}& 5.3578547x_3^2 + 0.8356891x_1x_5\\
&+ 37.293239x_1 - 40792.141
\end{aligned} \tag{22}$$

subject to

$$\begin{aligned}
g_1(\boldsymbol{x}) &= 85.334407 + 0.0056858x_2x_5\\
&\quad + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0\\
g_2(\boldsymbol{x}) &= -85.334407 - 0.0056858x_2x_5\\
&\quad - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0\\
g_3(\boldsymbol{x}) &= 80.51249 + 0.0071317x_2x_5\\
&\quad + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0\\
g_4(\boldsymbol{x}) &= -80.51249 - 0.0071317x_2x_5\\
&\quad - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0\\
g_5(\boldsymbol{x}) &= 9.300961 + 0.0047026x_3x_5\\
&\quad + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0\\
g_6(\boldsymbol{x}) &= -9.300961 - 0.0047026x_3x_5\\
&\quad - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0
\end{aligned} \tag{23}$$

where $78 \le x_1 \le 102$, $33 \le x_2 \le 45$ and $27 \le x_i \le 45$ $(i = 3, 4, 5)$. The optimum solution is $\boldsymbol{x}^*$ = (78, 33, 29.995256025682, 45, 36.775812905788) where $f(\boldsymbol{x}^*) = -30665.539$. Two constraints are active (g_1 and g_6).

E. `g05`

Minimize [9]

$$f(\boldsymbol{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3 \tag{24}$$

subject to

$$\begin{aligned}
g_1(\boldsymbol{x}) &= -x_4 + x_3 - 0.55 \le 0\\
g_2(\boldsymbol{x}) &= -x_3 + x_4 - 0.55 \le 0\\
h_3(\boldsymbol{x}) &= 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25)\\
&\quad + 894.8 - x_1 = 0\\
h_4(\boldsymbol{x}) &= 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25)\\
&\quad + 894.8 - x_2 = 0\\
h_5(\boldsymbol{x}) &= 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25)\\
&\quad + 1294.8 = 0
\end{aligned} \tag{25}$$

where $0 \le x_1 \le 1200$, $0 \le x_2 \le 1200$, $-0.55 \le x_3 \le 0.55$, and $-0.55 \le x_4 \le 0.55$. The best known solution [14] $\boldsymbol{x}^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ where $f(\boldsymbol{x}^*) = 5126.4981$.

F. g06

Minimize [6]

$$f(\boldsymbol{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{26}$$

subject to

$$\begin{aligned} g_1(\boldsymbol{x}) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0 \\ g_2(\boldsymbol{x}) &= (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0 \end{aligned} \tag{27}$$

where $13 \le x_1 \le 100$ and $0 \le x_2 \le 100$. The optimum solution is $\boldsymbol{x}^*$ = (14.095, 0.84296) where $f(\boldsymbol{x}^*) = -6961.81388$. Both constraints are active.

G. g07

Minimize [9]

$$\begin{aligned} f(\boldsymbol{x}) = {} & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\ & + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ & + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 \\ & + (x_{10} - 7)^2 + 45 \end{aligned} \tag{28}$$

subject to

$$\begin{aligned} g_1(\boldsymbol{x}) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0 \\ g_2(\boldsymbol{x}) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0 \\ g_3(\boldsymbol{x}) &= -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0 \\ g_4(\boldsymbol{x}) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0 \\ g_5(\boldsymbol{x}) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0 \\ g_6(\boldsymbol{x}) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0 \\ g_7(\boldsymbol{x}) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0 \\ g_8(\boldsymbol{x}) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0 \end{aligned} \tag{29}$$

where $-10 \le x_i \le 10$ $(i = 1, \cdots, 10)$. The optimum solution is $\boldsymbol{x}^*$= (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927) where g07 $(\boldsymbol{x}^*)$ = 24.306 209 1. Six constraints are active (g_1, g_2, g_3, g_4, g_5, and g_6).

H. g08

Minimize [14]

$$f(\boldsymbol{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \tag{30}$$

subject to

$$\begin{aligned} g_1(\boldsymbol{x}) &= x_1^2 - x_2 + 1 \le 0 \\ g_2(\boldsymbol{x}) &= 1 - x_1 + (x_2 - 4)^2 \le 0 \end{aligned} \tag{31}$$

where $0 \le x_1 \le 10$ and $0 \le x_2 \le 10$. The optimum is located at $\boldsymbol{x}^*$ = (1.2279713, 4.2453733) where $f(\boldsymbol{x}^*)$ = 0.095825. The solution lies within the feasible region.

I. g09

Minimize [9]

$$\begin{aligned} f(\boldsymbol{x}) = {} & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\ & + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \end{aligned} \tag{32}$$

subject to

$$\begin{aligned} g_1(\boldsymbol{x}) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0 \\ g_2(\boldsymbol{x}) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0 \\ g_3(\boldsymbol{x}) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0 \\ g_4(\boldsymbol{x}) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0 \end{aligned} \tag{33}$$

where $-10 \le x_i \le 10$ for $(i = 1, \cdots, 7)$. The optimum solution is $\boldsymbol{x}^*$ = (2.330499, 1.951372, −0.4775414, 4.365726, −0.6244870, 1.038131, 1.594227) where $f(\boldsymbol{x}^*)$ = 680.6300573. Two constraints are active (g_1 and g_4).

J. g10

Minimize [9]

$$f(\boldsymbol{x}) = x_1 + x_2 + x_3 \tag{34}$$

subject to

$$\begin{aligned} g_1(\boldsymbol{x}) &= -1 + 0.0025(x_4 + x_6) \le 0 \\ g_2(\boldsymbol{x}) &= -1 + 0.0025(x_5 + x_7 - x_4) \le 0 \\ g_3(\boldsymbol{x}) &= -1 + 0.01(x_8 - x_5) \le 0 \\ g_4(\boldsymbol{x}) &= -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \le 0 \\ g_5(\boldsymbol{x}) &= -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \le 0 \\ g_6(\boldsymbol{x}) &= -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \le 0 \end{aligned} \tag{35}$$

where $100 \le x_1 \le 10000$, $1000 \le x_i \le 10000$ $(i = 2, 3)$, and $10 \le x_i \le 1000$ $(i = 4, \cdots, 8)$. The optimum solution is $\boldsymbol{x}^*$ = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979) where $f(\boldsymbol{x}^*)$ = 7049.3307. Three constraints are active (g_1, g_2, and g_3).

K. g11

Minimize [14]

$$f(\boldsymbol{x}) = x_1^2 + (x_2 - 1)^2 \tag{36}$$

subject to

$$h(\boldsymbol{x}) = x_2 - x_1^2 = 0 \tag{37}$$

where $-1 \le x_1 \le 1$ and $-1 \le x_2 \le 1$. The optimum solution is $\boldsymbol{x}^* = (\pm 1/\sqrt{2}, 1/2)$ where $f(\boldsymbol{x}^*)$ = 0.75.

L. g12

Maximize [14]

$$f(\boldsymbol{x}) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100 \tag{38}$$

subject to

$$g(\boldsymbol{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0 \tag{39}$$

where $0 \le x_i \le 10$ $(i = 1, 2, 3)$ and $p, q, r = 1, 2, \cdots, 9$. The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that the above inequality holds. The optimum is located at $\boldsymbol{x}^*$ = (5,5,5) where $f(\boldsymbol{x}^*)$ = 1. The solution lies within the feasible region.

M. g13

Minimize [9]

$$f(\boldsymbol{x}) = e^{x_1x_2x_3x_4x_5} \tag{40}$$

subject to:

$$\begin{aligned} h_1(\boldsymbol{x}) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ h_2(\boldsymbol{x}) &= x_2 x_3 - 5 x_4 x_5 = 0 \\ h_3(\boldsymbol{x}) &= x_1^3 + x_2^3 + 1 = 0 \end{aligned} \tag{41}$$

where $-2.3 \leq x_i \leq 2.3$ $(i = 1, 2)$ and $-3.2 \leq x_i \leq 3.2$ $(i = 3, 4, 5)$. The optimum solution is $\boldsymbol{x}^*$ = (−1.717143, 1.595709, 1.827247, −0.7636413, −0.763645) where $f(\boldsymbol{x}^*)$ = 0.0539498.

ACKNOWLEDGMENT

The authors are grateful to Prof. Z. Michalewicz for answering their questions about his papers on constrained optimization, and to the three anonymous reviewers and Dr. D. Fogel for their constructive comments which helped to improve the clarity of this paper greatly.

REFERENCES

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford Univ. Press, 1996.

[2] E. Camponogara and S. N. Talukdar, "A genetic algorithm for constrained and multiobjective optimization," in *Proc. 3rd Nordic Workshop Genetic Algorithms and Their Appl. (3NWGA)*, J. T. Alander, Ed. Vaasa, Finland: Univ. Vaasa, Aug. 1997.

[3] A. R. Colville, "A comparison study of nonlinear programming codes," in *Princeton Symp. Math. Prog.*, H. W. Kuhn, Ed. Princeton, NJ: Princeton Univ. Press, 1970.

[4] K. Deb, "An efficient constrained handling method for genetic algorithms," in *Computer Methods in Applied Mechanics and Engineering*, 1999.

[5] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. New York: Wiley, 1968.

[6] C. Floundas and P. Pardalos, *A Collection of Test Problems for Constrained Global Optimization*. Berlin, Germany: Springer-Verlag, 1987, vol. 455, Lecture Notes in Comput. Sci..

[7] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*. New York: Wiley, 1997.

[8] D. Himmelblau, *Applied Nonlinear Programming*. New York: McGraw-Hill, 1972.

[9] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*. Berlin, Germany: Springer-Verlag, 1981, Lecture Notes in Econ. and Math. Syst..

[10] A. Homaifar, S. H.-Y. Lai, and X. Qi, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.

[11] F. Jiménez and J. L. Verdegay, "Evolutionary techniques for constrained optimization problems," in *Proc. 7th Eur. Congr. Intelligence Techniques and Soft Computing (EUFIT'99)*. Berlin, Germany: Springer-Verlag, 1999.

[12] J. Joines and C. Houck, "On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GAs," in *Proc. IEEE Int. Conf. Evolutionary Computing*. Piscataway, NJ: IEEE, 1994, pp. 579–584.

[13] S. Kazarlis and V. Petridis, "Varying fitness functions in genetic algorithms: Studying the rate of increase in the dynamic penalty terms," in *Parallel Problem Solving from Nature*. Berlin, Germany: Springer, 1998, vol. 1498, Lecture Notes in Comput. Sci., pp. 211–220.

[14] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evol. Comput.*, vol. 7, no. 1, pp. 19–44, 1999.

[15] Z. Michalewicz, "Genetic algorithms, numerical optimization and constraints," in *Proc. 6th Int. Conf. Genetic Algorithms*, L. J. Eshelman, Ed. San Mateo, CA: Morgan Kaufman, July 1995, pp. 151–158.

[16] Z. Michalewicz and N. Attia, "Evolutionary optimization of constrained problems," in *Proc. 3rd Annu. Conf. Evolutionary Programming*, A. V. Sebald and L. J. Fogel, Eds. River Edge, NJ: World Scientific, 1994, pp. 98–108.

[17] Z. Michalewicz, G. Nazhiyath, and M. Michalewicz, "A note on usefulness of geometrical crossover for numerical optimization problems," in *Proc. 5th Annu. Conf. Evolutionary Programming*, L. J. Fogel, P. J. Angeline, and T. Bäck, Eds. Cambridge, MA: M.I.T. Press, 1996, pp. 305–312.

[18] A. Petrowski and S. B. Hamida, "A logarithmic mutation operator to solve constrained optimization problems," in *Proc. 1999 Genetic and Evol. Comput. Conf.*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Hanavar, M. Jakiela, and R. E. Smith, Eds. San Mateo, CA: Morgan Kaufmann, 1999, p. 805.

[19] C. R. Reeves, "Genetic algorithms for the operations researcher," *INFORMS J. Comput.*, vol. 9, no. 3, pp. 231–247, 1997.

[20] H.-P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.

[21] W. Siedlecki and J. Sklansky, "Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition," in *Int. Conf. Genetic Algorithms*, 1989, pp. 141–149.

[22] A. E. Smith and D. W. Coit, "Penalty functions," in *Handbook on Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Oxford, U.K.: Oxford Univ. Press, 1997, pp. C5.2:1–C5.2:6.

[23] P. D. Surry and N. J. Radcliffe, "The COMOGA method: Constrained optimization by multiobjective genetic algorithms," *Contr. Cybern.*, vol. 26, no. 3, 1997.

[24] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 67–82, Apr. 1997.