Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization*

Abdel-Rahman Hedar and Masao Fukushima

Department of Applied Mathematics and Physics,

Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

April 21, 2004

Abstract

In this paper, a simulated-annealing-based method called Filter Simulated Annealing (FSA) method is proposed to deal with the constrained global optimization problem. The considered problem is reformulated so as to take the form of optimizing two functions; the objective function and the constraint violation function. Then, the FSA method is applied to solve the reformulated problem. The FSA method invokes a multi-start diversification scheme in order to achieve an efficient exploration process. To deal with the considered problem, a filter-set-based procedure is built in the FSA structure. Finally, an intensification scheme is applied as a final stage of the proposed method in order to overcome the slow convergence of SA-based methods. The computational results obtained by the FSA method are promising and show a superior performance of the proposed method, which is a point-to-point method, against population-based methods.

Key words: Constrained global optimization, Metaheuristics, Simulated annealing, Filter Set, Approximate descent direction

1 Introduction

Simulated Annealing (SA) is one of the most applicable metaheuristics in the optimization community. One of the most powerful features of SA is its ability of escaping from being trapped in local minima by accepting up-hill moves through a probabilistic procedure especially in the earlier stages of the search. For the continuous unconstrained optimization problem, SA has been intensively studied in its both forms; pure methods, see [1] and references therein, and hybrid methods, see [10, 13] and references therein. However, implementing SA on the continuous constrained optimization problem is still very limited in comparison with some other metaheuristics like the Evolutionary Algorithms (EAs). The SA approaches for constrained global optimization problems have been proposed by Wah,

*This research was supported in part by a Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science.

Wang and Chen, see [3, 31, 32, 33]. Another SA approach has been proposed by Romeijn and Smith [28]. These approaches are regarded as a pure SA. In this paper we propose a hybrid SA approach which invokes some intelligent concepts from other metaheuristics and local search methods. Specifically, we propose a SA-based approach called Filter Simulated Annealing (FSA) method for the following general nonlinear programming problem in the continuous space:

$$\begin{array}{ll} \min\limits_{x} & f(x), \\ \text{s.t.} & g_i(x) \leq 0, \; i = 1, \ldots, l, \\ & h_j(x) = 0, \; j = 1, \ldots, m, \\ & x \in \mathcal{S}, \end{array} \tag{P}$$

where f , g_i and h_j are real-valued functions defined on the search space $\mathcal{S} \subseteq R^n$. Usually, the search space $\mathcal{S}$ is defined as $\{x \in R^n : x_i \in [l_i, u_i], \; i = 1, \ldots, n\}$. The feasible region defined by the constraints is denoted by $\mathcal{F} \subseteq \mathcal{S}$.

Most of the metaheuristics which have been proposed to solve problem (P) are Evolutionary Algorithms (EAs). The optimization methods can be classified in two categories; the point-to-point methods which SA belongs to, and the population-based methods which EAs belong to. Metaheuristics from both categories have been successfully applied to the continuous unconstrained optimization problem. For example, some population-based methods such as genetic algorithms and scatter search methods have been proposed, see [11, 21] and references therein. As to point-to-point methods, tabu search, as well as SA, has also been invoked to deal with the continuous unconstrained optimization problem, see [12] and references therein. However, invoking point-to-point methods to deal with continuous constrained optimization problems is still very limited in comparison with the population-based methods. The main reason for unpopularity of SA for constrained global optimization problems, as well as most of the point-to-point methods, is its difficulty in keeping diversity. Especially, when the feasible region consists of several disjointed sub-regions, it is not so easy for a point-to-point method without a guidance of a diversification scheme to explore such regions effectively. Moreover, the point-to-point methods can be divided in two classes; single-start methods and multi-start methods. The later methods have shown efficient performance when applied to difficult optimization problems [23, 24, 30]. The standard SA belongs to the class of single-start methods. Therefore, there is a need to modify the standard SA in order to obtain an efficient method that can deal with the general case of problem (P).

In order to compose a powerful point-to-point-based method for solving problem (P), it is highly needed to consider the following things:

- In order to achieve efficient exploration of the space of interest, the designed method should consist of multi-start stages with a guidance of an effective diversification scheme. Otherwise, in the case of having disjointed feasible sub-regions, the method may be trapped in the first hit feasible sub-region.

- An efficient exploration process should also invoke a search procedure which has the ability to explore both of feasible and infeasible regions, rather than exploring the feasible region only. This is needed to reach the global solution specially in the following cases:

- if the global solution lies on the boundary between feasible and infeasible regions,
- if the global solution lies in a feasible sub-region which differs from the currently considered sub-region in the case of having several disjointed feasible sub-regions.

- In the most cases of constrained optimization problems like problem (P), optimal solutions usually lie on the boundary of the feasible region. In order to explore the region near the boundary between the feasible and infeasible regions effectively, the designed method should invoke a solution generation procedure which is able to intensify the solution generation process.

- An elite-based intensification scheme is needed in the final stage in order to refine the best solution found so far. Especially, if the method is SA-based, a quicker intensification scheme is highly needed to overcome the slowness of SA in its final stage.

We have considered all the above in designing the FSA method. So the FSA method is a multi-start method with a diversification scheme. The FSA method uses the filter set concept [7] in accepting new trial solutions, which gives it the ability to explore both of feasible and infeasible regions. Moreover, the FSA method generates more trial solutions whenever the region near the boundary is reached. Finally, two types of intensification schemes are applied in order to refine the best solution visited so far. Thus, the FSA method is a hybrid method which takes advantage of low computational cost of point-to-point methods and efficient exploration of population-based methods. In other words, the FSA method is an attempt to design a point-to-point method that behaves like a population-based method without spending a big computational cost.

The numerical results shown later indicate that the proposed FSA method is very promising in the quality of obtained solutions as well as the computational costs especially for dealing with constraints. Moreover, the numerical results also show that the FSA method is competitive with the population-based methods in the quality of solution and it is much cheaper than them in the computational costs. In the next section, we give some preliminaries needed throughout the paper. In Section 3, we highlight the main components of the proposed FSA method. The study of the FSA parameters is given in Section 4. In Sections 5 and 6, we report numerical results for the FSA method. Finally, the conclusion makes up Section 7.

2 Preliminaries

This section highlights the idea of reformulating problem (P) as a multiobjective optimization problem and the concepts of filter set and filtered points. To achieve that, the concept of Pareto dominance in multiobjective optimization should be defined first.

2.1 Pareto Dominance

Pareto Dominance is the most common concept of optimality in the multiobjective optimization field. Multiobjective optimization seeks to optimize a vector of objective functions within a feasible decision variable space. For the multiobjective minimization problem with the objective functions $\varphi_1(x), \ldots, \varphi_q(x)$, defined on the search space $\mathcal{S}_M \subseteq R^n$, the Pareto Dominance is defined as follows:

Definition 1 *An objective vector $\Phi(y) = (\varphi_1(y), \ldots, \varphi_q(y))$ is said to dominate another objective vector $\Phi(z) = (\varphi_1(z), \ldots, \varphi_q(z))$, written $\Phi(y) \prec \Phi(z)$, if and only if $\varphi_i(y) \leq \varphi_i(z), \forall i = 1, \ldots, q$ and there exists at least one $j \in \{1, \ldots, q\}$ such that $\varphi_j(y) < \varphi_j(z)$.*

We will write $\Phi(y) \preceq \Phi(z)$ to indicate that either $\Phi(y) \prec \Phi(z)$ or $\Phi(y) = \Phi(z)$. In the rest of the paper, we will simply write $y \prec z$ and $y \preceq z$ instead of writing $\Phi(y) \prec \Phi(z)$ and $\Phi(y) \preceq \Phi(z)$, respectively.

2.2 Problem Reformulation

An effective approach to handle constraints is to use multiobjective optimization techniques, see for example [4, 5]. Such approaches reformulate the constrained problem as a multiobjective problem involving the original objective function and constraint violation functions. More specifically, by introducing the constraint violation functions

$$\begin{aligned} G_i(x) &= (\max[0, g_i(x)])^\alpha, \; i = 1, \ldots, l, \\ G_{l+j}(x) &= |h_j(x)|^\alpha, \; j = 1, \ldots, m, \end{aligned} \tag{1}$$

where α is normally chosen to be 1 or 2, problem (P) can be reformulated as the following multiobjective optimization problem:

$$\min_{x \in \mathcal{S}} \left[f(x), G_1(x), \ldots, G_{m+l}(x)\right]. \tag{P_M}$$

Alternatively, we may consider another reformulation of problem (P) as the following bi-objective optimization problem:

$$\min_{x \in \mathcal{S}} \left[f(x), G(x)\right], \tag{P_B}$$

where $G(x) = \sum_{i=1}^{m+l} G_i(x)$. The method proposed in this paper will deal with problem (P) through the reformulated problem (P_B). In particular, we will denote $x \prec y$ if x dominates y with respect to the vector function $\Phi(x) = (f(x), G(x))$.

2.3 Filter Set and Filtered Points

The *filter set* $\mathfrak{F}$ is defined as a finite set of infeasible[1] points in $\mathcal{S}$ such that $x \prec y$ does not hold for any x and y in $\mathfrak{F}$. The point x^F with the minimum function value $f(x)$ found so far in the feasible region $\mathcal{F} = \{x \in \mathcal{S} : G(x) = 0\}$ is saved and treated separately as a single filter point. This definition is taken from [2] which differs slightly from the original definition in [7]. A point y is called a *filtered point* [2], if one of the following holds:

[1]Throughout the paper, the feasibility is related only to problem (P) rather than P_M or P_B, that is, we call a point $x \in \mathcal{S}$ feasible if $G(x) = 0$, and infeasible if $G(x) > 0$.

- $y \succeq x$ for some $x \in \mathfrak{F}$.
- $G(y) \geq G_{\max}$, where $G_{\max} > 0$ is maximum value allowed on the constraint violation function $G(x)$.
- $G(y) = 0$, and $f(y) \geq f^F$, where $f^F = f(x^F)$ is the minimum function value found so far in the feasible region.

In other words, we have three kinds of filtered point sets:

$$\begin{aligned}
\bar{\mathfrak{F}}_{\mathrm{I}} &= \{y \in \mathcal{S} : y \succeq x \text{ for some } x \in \mathfrak{F}\}, \\
\bar{\mathfrak{F}}_{\mathrm{II}} &= \{y \in \mathcal{S} : G(y) \geq G_{\max}\}, \\
\bar{\mathfrak{F}}_{\mathrm{III}} &= \left\{y \in \mathcal{S} : G(y) = 0,\ f(y) \geq f^F\right\}.
\end{aligned}$$

Therefore, the set of all filtered points is defined as $\bar{\mathfrak{F}} = \bar{\mathfrak{F}}_{\mathrm{I}} \cup \bar{\mathfrak{F}}_{\mathrm{II}} \cup \bar{\mathfrak{F}}_{\mathrm{III}}$. Unfiltered points are used to update $\bar{\mathfrak{F}}$ by adding them and deleting the old ones which are dominated by the new added points.

3 The FSA method

The FSA method starts with a diversification generation procedure to generate a set of diverse solutions called *DivSet*. The initial solution is chosen from the *DivSet*. Then, the *DivSet* stands by to provide the search with a diverse solution whenever further diversification is needed. In the FSA method, we introduce a ranking procedure for comparing and ordering solutions. This ranking procedure is based on the filter set as well as objective function and constraint violation function values. The scenario in the FSA method can be described as follows. Let the current trial solution be *Sol*. Using the FSA ranking procedure, *Sol* is initialized to be the best ranked one in *DivSet*. Then, trial solutions are generated in a neighborhood of *Sol* using a trial solution generation procedure based on the approximate descent direction (ADD) method proposed in [13]. The trial solution generation procedure generates trial solutions in such a way that the objective function value is likely to decrease if *Sol* is feasible, and the constraint violation function value is likely to decrease if *Sol* is infeasible. Moreover, the trial solution generation procedure intensifies the solution generation process if *Sol* is close to the boundary of the feasible region. We try to update *Sol* with one of the generated trial solutions using the simulated annealing acceptance concept. Specifically, if an unfiltered trial solution is obtained, we accept it with probability 1. Otherwise, a trial solution is accepted with a certain probability controlled by the temperature parameter. Whenever the number of consecutive iterations without accepting a new trial solution exceeds a predetermined maximum number, a new diverse solution is chosen from *DivSet* and the re-annealing process is applied, i.e., the temperature is re-initialized. While the search proceeds, *DivSet* is updated by removing any of its elements if the search reaches a region close to this element. We terminate this main stage of the FSA method when the cooling schedule is completed with an empty *DivSet*. Finally, two intensification schemes are invoked to refine the best solution found so far. The best solution is defined to be the best feasible solution if the feasible region is reached. Otherwise, the best solution is defined to be the infeasible solution with the least constraint violation function value. The temperature parameter at the best solution found so far in the previous search stage is saved to be used in the first intensification scheme which applies an annealing process with slower cooling schedule and smaller step sizes. The second intensification scheme applies a greedy local search method on a penalty function of problem (P) starting from the best solution found so far. Figure 1 shows the outline of the FSA method. Below, we describe the details of the FSA main steps sketched above and state the FSA algorithm formally at the end of this section.

3.1 Diversification Generation Procedure

In the FSA method, we use the scatter search diversification generation Method [21, 22] to generate a diverse solution set *DivSet.* In that method, the interval $(u_i - l_i)$ of each variable is divided into 4 sub-intervals of equal size. For each sub-interval of each variable, a *frequency count* is defined as the number of solutions which are perviously chosen in this sub-interval. To generate a new solution to be added to *DivSet*, one has to

- choose one sub-interval for each variable randomly with a probability inversely proportional to its frequency count, and
- choose a random value for each variable that lies in the corresponding selected sub-interval.

While the search proceeds, the *DivSet* is updated by eliminating any of its elements lying close to a visited solution. Specifically, when the current solution is x, the *DivSet* is updated through the rule

$$DivSet = DivSet - \{y : y \in DivSet,\ \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{H_i^2} \leq 1\}, \tag{2}$$

where $H_{Div} = (H_1, \ldots, H_n)$ is a predetermined constant vector with positive components.

When the diversification is needed, the solution with the largest distance from the current solution is chosen from the *DivSet* to be a new diverse solution.

3.2 Ranking Procedure

To order the solutions in a set $S = \{x_1, x_2, \ldots, x_\mu\}$, we introduce the following ranking procedure. The solutions are ordered based on three rank functions as given below.

1. *Dominance Rank* (r_d): The best feasible point x^F is given the rank value $r_d = 1$, and other feasible points are given the rank value $r_d = 2$. The points in $\mathfrak{F}$ are given the rank value $r_d = 1$, and any other infeasible point x is given the rank value $r_d = \nu + 1$, where ν is the number of points in $\mathfrak{F}$ which dominate x.
2. *f-value Rank* (r_f): According to their objective function values $f(x_i),\ x_i \in S$, the best point is given the rank value $r_f = 1$, the second best point is given the rank value $r_f = 2$, and so on.

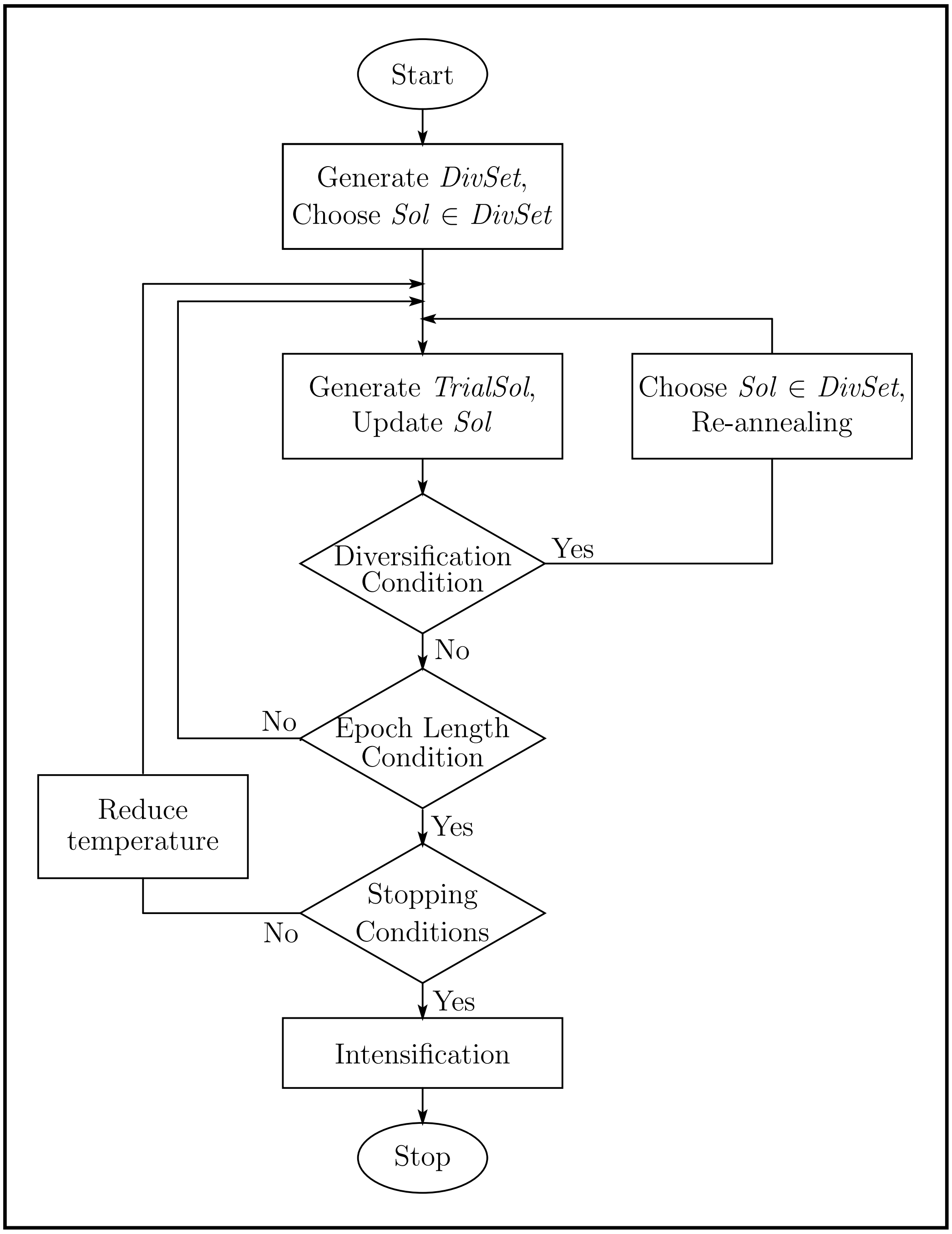

Figure 1: Outline of the FSA method.

3. *G-value Rank* (r_G): According to their constraint violation function values $G(x_i)$, $x_i \in S$, the best point is given the rank value $r_G = 1$, the second best point is given the rank value $r_G = 2$, and so on.

In each ranking described above, ties are broken arbitrarily. Then, the total ranking function r is defined by

$$r(x_i) = r_d(x_i) + \frac{\lambda}{\mu} r_f(x_i) + \frac{(1-\lambda)}{\mu} r_G(x_i), \; x_i \in S, \tag{3}$$

where $\lambda \in [0, 1]$. The solutions of S are ordered and relabeled such that

$$r(x_1) \leq r(x_2) \leq \cdots \leq r(x_{n+1}). \tag{4}$$

The main role of the parameter λ is to control the priority in the ranking between the objective function value and the feasibility. Actually, the ranking function r is basically based on the dominance rank r_d and, within the same dominance rank value, the parameter λ gives a greater value to either of the ranking values r_f and r_G. Specifically, setting $\lambda \in [0, 0.5)$ gives some priority to the feasible points and setting $\lambda \in (0.5, 1]$ gives some priority to points with lower objective function values. In the FSA method, the value of λ is chosen to be less than $1/\mu$ in order to accept a better feasible solution when it is found. Moreover, this ranking procedure allows a new infeasible solution which reduces the constraint violation function to be highly accepted, compared with a new feasible solution which is worse than the best feasible solution found so far. This gives the search process more flexibility to explore the boundary region.

3.3 Trial Solution Generation Procedure

We use Approximate Descent Direction (ADD) method [13] to generate trial solutions in the FSA method. So we first summarize the ADD method before stating the trial solution generation procedure. The ADD method is a derivative-free procedure with high ability of producing a descent direction. In the ADD method, several points around a given point $x \in R^n$ are used to generate an approximate descent direction of a function ψ at x. More specifically, the ADD method chooses p points close to x, called exploring points, in order to generate an approximate descent direction $v \in R^n$ of ψ at x, where p is some positive integer. The exploring points, say $\{y_i\}_{i=1}^p$, are used to compute the direction v as follows:

$$v = \sum_{i=1}^{p} w_i e_i, \tag{5}$$

where

$$\begin{aligned} w_i &= \frac{\Delta\psi_i}{\sum_{j=1}^p |\Delta\psi_j|}, \quad i = 1, 2, \ldots, p, \\ e_i &= -\frac{y_i - x}{\|y_i - x\|}, \quad i = 1, 2, \ldots, p, \\ \Delta\psi_i &= \psi(y_i) - \psi(x), \quad i = 1, 2, \ldots, p. \end{aligned}$$

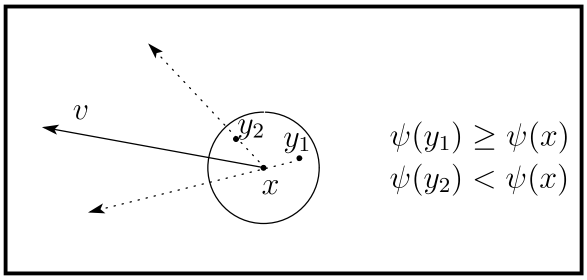

Figure 2: An ADD example in R^2.

By means of (5), the direction v is composed toward the vectors $-sign\,(\Delta\psi_i)\,(y_i - x)$ with weights proportional to $|\Delta\psi_i|$, $i = 1, 2, \ldots, p$. Figure 2 shows an example of composing an ADD in two dimensions. Given a point $x \in R^2$, the ADD v is composed in Figure 2 toward

- the vector $-(y_1 - x)$, since the inequality $\psi(y_1) \geq \psi(x)$ suggests that the function value is not likely to decrease along the direction $y_1 - x$, and
- the vector $y_2 - x$, since the inequality $\psi(y_2) < \psi(x)$ suggests that the function value is likely to decrease along the direction $y_2 - x$.

In the FSA method, we use the ADD method to generate a search direction d at a given solution x, and then use it to generate new trial solutions in a neighborhood of x. Specifically, we first generate p exploring points close to x and generate a search direction as follows:

1. If x is feasible, we apply the ADD method using the generated exploring points to compute an approximate descent direction v_f of f at x. Then, we set the search direction $d := v_f / \|v_f\|$.
2. If x is infeasible, we apply the ADD method using the generated exploring points to compute an approximate descent direction v_G of G at x. Then, we set the search direction $d := v_G / \|v_G\|$.

Trial solutions can be generated along the search direction d with suitable step sizes. Moreover, it is known that, in most cases, optimal solutions can be found on the boundary of the feasible region. So, in order to encourage the search to explore the region near the boundary effectively, more trial solutions should be generated whenever the current solution is close to the boundary. To implement this idea in the FSA method, another trial solution will be generated between the current solution and the trial solution if the feasibility status is changed between them. Figure 3 shows an example of the two types of generating trial solutions in the neighborhood of a current solution x. In Figure 3(a), a trial solution y is generated along the search direction d and since x and y are feasible, no more trial solution will be generated. However, in Figure 3(b), x is feasible but y is infeasible, and so another trial solution y' is generated between x and y. Formally, we can define the trial solution set as

$$TS(x) = \{y : y = x + \delta_i \Delta d,\ i \in I\}, \tag{6}$$

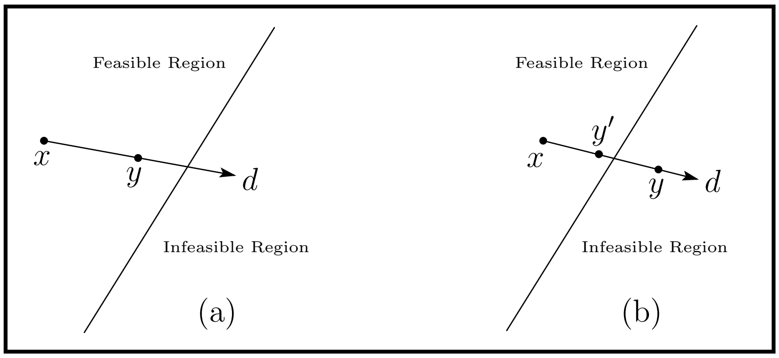

Figure 3: An example of generating trial solutions.

where Δ is a step size and δ_i are random numbers. The set I is given by $I = \{1\}$ if the feasibility status at $x + \delta_1 \Delta d$ is the same as that at x, i.e., $G(x + \delta_1 \Delta d) = G(x) = 0$, or $G(x + \Delta d) > 0$ and $G(x) > 0$, and $I = \{1, 2\}$, otherwise. The random numbers δ_i give the search some stochastic behavior to achieve more efficient exploration. For example, we may let δ_1 be uniformly distributed in the interval $(0, 1)$ and δ_2 be normally distributed with mean $1/2$ and a suitable variance σ^2.

3.4 Intensification

In the FSA method, we compose two stages of intensification process. The first one is SA-based procedure, called *SA Intensification*, in which up-hill movements may be accepted in order to avoid the case where the region around the best solution visited so far is prematurely explored in the previous search stages. The other stage of intensification is a greedy process that accepts only down-hill movements, which we call *Local Search Intensification.* This greedy-type intensification is needed since it has been reported that the SA can reach a region near global minima, however, it may wander around the optimal solution if high accuracy is required [13, 34]. The outline of these intensification stages is given below.

- **SA Intensification.** In the previous stage of the search, we save the temperature parameter value recorded at the best solution found so far. Then, in order to refine that solution, a slower cooling schedule, i.e., a schedule with a higher cooling ratio, is started from the saved value of the temperature parameter. Moreover, the step size used in generating trial solutions is reduced to refine the search steps for more accurate exploration.

- **Local Search Intensification.** A direct search method is applied, starting from the best solution found so far, to minimize the penalty function

$$p(x) = f(x) + \rho G(x), \tag{7}$$

  where ρ is a penalty parameter. The Kelley's modification [15, 16] of the Nelder-Mead method [27] is used to minimize the function $p(x)$ in N consecutive times using gradually increasing penalty parameters $\rho_1, \rho_2, \ldots, \rho_N$.

3.5 FSA Algorithm

The formal description of the FSA method is given below.

Algorithm FSA

1. Initialization. Construct *DivSet* using the diversification generation procedure. Set the best ranked point in *DivSet* to be the initial point x_0. Choose the cooling schedule parameters; initial temperature T_{max}, final temperature T_{min} and cooling ratio $\gamma \in (0, 1)$, and the epoch length M and set $T := T_{max}$. Set $\mathfrak{F}_0$ to be empty, set $x^{best} = x_0$, choose a step size $\Delta > 0$, choose a positive integer $K_{\max}$, and set $k := 0$.

2. Main Loop.

2.1. Compute a trial solution set $TS(x_k)$ as in (6) with the step size Δ. Set y_k equal to the best ranked point in $TS(x_k)$.

2.2. The trial point y_k is accepted with the probability

$$p = \begin{cases} 1, & \text{if } y_k \notin \bar{\mathfrak{F}}_k, \\ \min\{1, \exp\left(-\Delta_{fG}/T\right)\}, & \text{otherwise}, \end{cases} \tag{8}$$

where $\Delta_{fG} = \max\left\{f\left(y_k\right) - f\left(x_k\right), G\left(y_k\right) - G\left(x_k\right)\right\}$.

2.3. If y_k is accepted, then set $x_{k+1} := y_k$; otherwise, set $x_{k+1} := x_k$. Update $\mathfrak{F}_k$, x^{best} and *DivSet*, and set $k := k + 1$.

2.4. *Diversification.* If the number of consecutive iterations without accepting a new solution exceeds $K_{\max}$, and *DivSet*$\neq \phi$, then choose $x_k \in$ *DivSet*, set $T := T_{max}$, set $\mathfrak{F}_k$ to be empty, and go to Step 2.1. Otherwise, go to Step 2.5.

2.5. If the epoch length M is attained, then go to Step 2.6. Otherwise, go to Step 2.1.

2.6. If $T > T_{min}$, then set $T := \gamma T$ and go to Step 2.1. Otherwise, go to Step 3.

3. Intensification.

3.1. *SA Intensification.* Set x_k equal to x^{best}, set T equal to the saved temperature at that point, and set a final temperature T'_{min}, an epoch length M' and a cooling ratio $\gamma' > \gamma$.

3.1.1 Compute a trial solution set $TS(x_k)$ as in (6) with the step size Δ. Set y_k equal to the best ranked point in $TS(x_k)$.

3.1.2 Accept y_k with the probability p given by (8). Set $x_{k+1} := y_k$ if y_k is accepted; otherwise, set $x_{k+1} := x_k$. Update $\mathfrak{F}_k$ and x^{best}, and set $k := k + 1$.

3.1.3 If the epoch length M' is attained, then go to Step 3.1.4. Otherwise, go to Step 3.1.1.

3.1.4 If $T > T'_{min}$, then set $T := \gamma' T$ and go to Step 3.1.1. Otherwise, go to Step 3.2.

3.2. *Local Search Intensification.* For $\rho = \rho_1, \rho_2, \ldots, \rho_N$:

3.2.1 Apply a local search method to the function $f(x) + \rho G(x)$ starting from x^{best}.

3.2.2 Update x^{best} and go to Step 3.2.1.

Table 1: The FSA parameters

Parameter Group	Parameter	Definition
Constraint Violation	α	Power factor used in (1)
Function	ϵ	Small positive number used for reformulating equality constraints
Diversification	$\lvert DivSet \rvert$	Size of *DivSet*
	H_{Div}	Distance vector used to update *DivSet*
	$K_{\max}$	Maximum number of iterations allowed without acceptance
Cooling Schedule	T_{max}, T_{min}	Initial and final temperatures
	M	Epoch length
	γ	Cooling ratio
Trial Solutions	p	Number of exploring points used in ADD
	r	Neighborhood radius used in ADD
	Δ	Step size used in (6)
	σ^2	Variance of the normal distribution of δ_2
	λ	Rank ordering parameter
	$G_{\max}$	Maximum value allowed on $G(x)$
Intensification	T'_{min}	Final temperature in SA Intensification
	M'	Epoch length
	γ'	Cooling ratio in SA Intensification
	$\rho_1, \rho_2, \ldots, \rho_N$	Penalty parameters

4 Setting FSA Parameters

In this section, setting the FSA parameters is discussed to complete the implementation of the FSA algorithm stated in the previous section. These parameters can be classified as shown Table 1, which contains all FSA parameters and their definitions. Some preliminary numerical experiments have been done in order to find proper values of these parameters. Moreover, these experiments of tuning parameters aim to obtain a standard setting of parameters which is problem-independent as much as possible. The values of some parameters are assigned to their standard setting reported in the literature. Below, we state the suggested values of the FSA parameters as well as the conclusion of what we got from the tuning parameters experiments.

4.1 Constraint Violation Function Parameters

The power factor α used in (1) is set equal to 2, since using this value showed notably better performance of the FSA method than that of using the value 1. Treating the equality constraints as in (1) does not seem efficient in the implementation. It was observed that reformulating the equality constraint $h(x) = 0$ as the inequality constraint $|h(x)| - \epsilon \leq 0$, where ϵ is a small positive number, yielded a better performance of the FSA method.

Moreover, using a large value of ϵ in the first stage of search and reducing its value in the intensification stage gave better results. Therefore, we set ϵ equal to 10^{-3} in reformulating all equality constraints in all FSA search stages except in the local search intensification stage in which ϵ is set equal to 10^{-6}.

4.2 Diversification Parameters

The size of *DivSet* depends on many factors such as the wideness of the search space, the number of disjoint feasible sub-regions, and the multimodality of the objective function. We observed that setting the size of *DivSet* equal to 50 almost fits all the considered problems. The distance vector $H_{Div} = (H_1, \dots, H_n)$ used to update the *DivSet* is set so as to fit the size of the search space. Specifically, we set $H_i = \frac{u_i - l_i}{|DivSet|/n}$, $i = 1, \dots, n$, where the denominator represents the average line density of distributing the solutions of *DivSet* along each coordinate direction, so that the value of H_i represents the average distance along the coordinate direction i between two neighboring diverse solutions. The maximum number $K_{\max}$ of iterations allowed without accepting new trial solutions is set equal to 10.

4.3 Cooling Schedule Parameters

The initial temperature $T_{\max}$ is set large enough to make the initial probability of accepting transition close to 1. Beside the initial point x_0, another point $\widetilde{x}_0$ is generated in a neighborhood of x_0 to calculate $T_{\max}$ as

$$T_{\max} := -\frac{1}{\ln(0.9)} \left| f(\widetilde{x}_0) - f(x_0) \right| .$$

At the beginning of each re-annealing process, a new $T_{\max}$ is computed in a similar manner. The cooling ratio γ is normally chosen to be between 0.9 and 0.99 [19]. In our experiments, we set γ equal to 0.9 and a higher value is used in the intensification stage as we will state later. A common choice of the epoch length M is to let it depend on the size of the problem [17, 20]. In our experiments, we set M equal to $2n$. The cooling schedule is terminated when the temperature reaches a fixed minimum value $T_{\min}$. We observed that setting $T_{\min}$ equal to $\min(10^{-5}, 10^{-5}T_{\max})$ could give a complete cooling schedule in the sense that the acceptance probability at the end is almost zero.

4.4 Trial Solutions Parameters

The parameters used in computing the search directions v_f and v_G are the number p of exploring points, and the radius r of the neighborhood in which the exploring points are generated. We set $p = 2$ and $r = 10^{-3}$ as suggested in [13]. The ranking parameter λ is set equal to $0.5/\mu$, where μ is the number of solutions to be ranked or compared. This setting allows the best feasible solution to have the highest rank, whenever it exists, among the compared solutions. Setting a proper value of step size Δ is very effective in the performance of the FSA algorithm, because setting too big a value for Δ may yield a premature termination of the algorithm and setting too small a value for Δ will not yield an efficient exploration process for the whole search space. We tested many values of Δ

and found that the value $\Delta = \min\left(0.05\sum_{i=1}^{n}(u_i - l_i)/n, 10\right)$ gave the best performance. For variance σ^2 of the normal distribution of δ_2, the values $\sigma = 1/2, 1/3, 1/4$ have been tested. We observed that setting $\sigma = 1/3$ gave a slightly better performance than setting the other values. The filter set contains only one parameter; the maximum value $G_{\max}$ allowed on the constraint violation function $G(x)$. In the original reference of the filter set concept [7], the value of $G_{\max}$ is set equal to $\max(1.25G(x_0), 100)$, where x_0 is the initial solution. However, in the FSA algorithm, we use a higher value, since our goal is to explore the whole search space effectively and reaching a global minimum, which differs from the goal of [7], i.e., finding a local minimum. So we set $G_{\max}$ equal to $10\max(1.25G_{\max}^{Div}, 100)$, where $G_{\max}^{Div}$ is the maximum value of the constraint violation function $G(x)$ computed at each point in the *DivSet*.

4.5 Intensification Parameters

For the SA Intensification Parameters, final temperature T'_{min}, epoch length M' and cooling ratio γ' are set equal to $10^{-5}T_{best}$, $2n$ and 0.99, respectively, where T_{best} is the temperature saved at the best solution found so far. The number N of times the local search method is applied in the local search intensification stage is set equal to 4. The penalty parameters used in these local searches are $\rho_1 = 10^{\beta+2}$, $\rho_1 = 10^{\beta+4}$, $\rho_1 = 10^{\beta+6}$ and $\rho_1 = 10^{\beta+10}$, where β is the number that appears in the floating point form $\alpha_1.\alpha_2\alpha_3\cdots\times 10^{\beta}$ of the best point found so far.

5 Numerical Results

In this section, we report the performance of the FSA algorithm on 13 well-known test problems G1–G13 [14, 18, 26], which are shown in the Appendix. The characteristics of those test problems are diverse enough to cover many kinds of difficulties that constrained global optimization problems face. More experimental results on three other application problems will be shown in the next section.

The FSA code was applied to solve each considered problem 30 times with different starting solutions. For all test problems, the values of the FSA parameters remained constant at those values which have been presented in the previous section. Table 2 summarizes the FSA results obtained for each test problem as well as the best known objective function value for each problem. Problems G2, G3 and G8 are maximization problems originally so that they were solved by converting them to minimization problems. In Table 2, the best and the worst objective function values obtained from 30 runs are reported for each test problem. In order to show more details concerning the quality of the obtained solutions, the average and the standard deviation of the obtained objective function values are also reported in Table 2. Moreover, the average numbers Av. f-evals. and Av. c-evals. of objective and constraint functions evaluations, respectively, are shown in the last two columns of Table 2. It is noteworthy that the FSA method is very economical in computing the constraint function values as shown in Table 2.

The results obtained by the FSA method are quite satisfactory, except for problem G2 which has the highest dimension among all test problems G1–G13. On the other hand, the

Table 2: FSA results for problems G1–G13

Pr.	Type	Best Known	Best	Av.	Worst	S.D.	Av. f-evals.	Av. c-evals.
G1	min	–15	–14.999105	–14.993316	–14.979977	0.004813	205,748	87,701
G2	max	0.803619	0.7549125	0.3717081	0.2713110	0.098023	227,832	101,903
G3*	max	1	1.0000015	0.9991874	0.9915186	0.001653	314,938	118,404
G4	min	–30665.539	–30665.5380	–30665.4665	–30664.6880	0.173218	86,154	37,000
G5*	min	5126.4981	5126.4981	5126.4981	5126.4981	0.000000	47,661	17,757
G6	min	–6961.81388	–6961.81388	–6961.81388	–6961.81388	0.000000	44,538	15,817
G7	min	24.3062091	24.310571	24.3795271	24.644397	0.071635	404.501	171,299
G8	max	0.095825	0.095825	0.095825	0.095825	0.000000	56,476	23,219
G9	min	680.6300573	680.63008	680.63642	680.69832	0.014517	324,569	147,035
G10	min	7049.3307	7059.86350	7509.32104	9398.64920	542.3421	243,520	93,667
G11*	min	0.75	0.7499990	0.7499990	0.7499990	0.000000	23,722	8,485
G12	min	–1	–1.0000000	–1.0000000	–1.0000000	0.000000	59,355	25,818
G13*	min	0.0539498	0.0539498	0.2977204	0.4388511	0.188652	120,268	42,268

* Problems contain equality constraints.

results for problem G12 are very promising since the feasible region of this problem consists of 9^3 disjointed spheres with radus 0.25. The FSA method could successfully find global minima in all runs with low computational costs as shown in Table 2. This indicates the success of the multi-start diversification scheme invoked in the FSA method. For problem G11, the FSA method reached a point with objective function value 0.7499990 for all 30 runs. However, by decreasing the parameter ϵ, which is used to convert the equality constraint to the inequality one, from 10^{-6} to 10^{-10} in the local search intensification, the FSA method easily reached the exact global minimum with objective function value 0.75 in all runs.

To complete examining the FSA performance, its results are compared with those of other EA-based methods proposed for dealing with problem (P). The EA-based methods that we used in the comparison are

1. Homomorphous Mappings (HM) method [18],
2. Stochastic Ranking (SR) method [29],
3. Adaptive Segregational Constraint Handling EA (ASCHEA) method [9],
4. Simple Multimembered Evolution Strategy (SMES) method [25].

The challenge that the FSA method faces is to what extent a point-to-point method behaves like a population-based method or even better. To examine this issue, two measurements are considered; solution qualities and computational costs. First, we discuss the solution qualities and, later at the end of this section, we will discuss computational costs. The results of the compared methods, which are taken from their original references [9, 18, 25, 29], as well as those of the FSA method are reported in Table 3 to show the solution qualities obtained by them. It is not easy to draw a definite conclusion from comparisons due to different accuracies used in the respective results. However, we state below some comments on the results reported in Table 3. All the results in Table 3 are obtained from 30 runs of each method except those of HM method, which are obtained from 20 runs. The HM method could obtain the optimal solution in all runs for only problem G11. For the other methods SR, ASCHEA, and SMES, they could obtain the optimal solutions in all runs for problems {G1,G3,G4,G8,G11,G12}, {G4,G6,G8,G11} and {G1,G4,G8,G12}, respectively. The FSA method could obtain the optimal solutions in all runs for problems {G5,G6,G8,G11,G12}. It is noteworthy that the FSA method could obtain the optimal solution in all runs for problem G5, whereas the other methods failed to obtain it even in a single run. Moreover, the FSA method could obtain the optimal solution of problem G13 in 7 out of 30 runs, while the other methods failed to obtain it.

The computational costs of the considered EA-based methods are extremely expensive compared with those of the FSA method. Since there is no automatic termination criteria for the considered EA-based methods, thy were terminated when the number of generations exceeds a predetermined maximum number. Therefore, the computational costs of these methods are problem-independent, i.e., the number of objective and constraint functions evaluation remains constant for each test problem. Specifically, computational costs for HM, SR, ASCHEA and SMES used in each test problem, which are taken from their original references [9, 18, 25, 29], are 1400000, 350000, 1500000 and 250000 fitness function evaluations, respectively, and each fitness function evaluation requires one evaluation of the objective function and one evaluation of each constraint function. The main reason for these high computational costs is that EAs are not equipped with automatic termination criteria and this is one main drawback of EAs. For some of the test problems, the considered EA-based methods could obtain an optimal solution in an early stage of the search, but they were not learned enough to judge whether they could terminate. On the other hand, the EA-based methods have less parameters than SA-based methods. However, in the FSA method as well as SA-based methods, some preliminary experiments on tuning parameters will let them learn applicable termination criteria.

6 More Numerical Experiments

In this section, we discuss the results of the FSA method on some application problems. Three problems from the engineering optimization area are considered.

Table 3: Results of FSA and other EA-based methods for problems G1–G13

Pr.	Type	Best Known		HM	SR	ASCHEA	SMES	FSA
G1	min	–15	Best	–14.7864	–15	–15	–15	–14.999105
			Av.	–14.7082	–15	–14.84	–15	–14.993316
			Worst	–14.6154	–15	N.A.	–15	–14.979977
G2	max	0.803619	Best	0.79953	0.803515	0.785	0.803601	0.7549125
			Av.	0.79671	0.781975	0.59	0.785238	0.3717081
			Worst	0.79119	0.726288	N.A.	0.751322	0.2713110
G3*	max	1	Best	0.9997	1.000	1	1.001038	1.0000015
			Av.	0.9989	1.000	0.99989	1.000989	0.9991874
			Worst	0.9978	1.000	N.A.	1.000579	0.9915186
G4	min	–30665.539	Best	–30664.5	–30665.539	–30665.5	–30665.539062	–30665.5380
			Av.	–30655.3	–30665.539	–30665.5	–30665.539062	–30665.4665
			Worst	–30645.9	–30665.539	N.A.	–30665.539062	–30664.6880
G5*	min	5126.4981	Best	–	5126.497	5126.5	5126.599609	5126.4981
			Av.	–	5128.881	5141.65	5174.492301	5126.4981
			Worst	–	5142.472	N.A.	5304.166992	5126.4981
G6	min	–6961.81388	Best	–6952.1	–6961.814	–6961.81	–6961.813965	–6961.81388
			Av.	–6342.6	–6875.940	–6961.81	–6961.283984	–6961.81388
			Worst	–5473.9	–6350.262	N.A.	–6961.481934	–6961.81388
G7	min	24.3062091	Best	24.620	24.307	24.3323	24.326715	24.310571
			Av.	24.826	24.374	24.6636	24.474926	24.3795271
			Worst	25.069	24.642	N.A.	24.842829	24.644397
G8	max	0.095825	Best	0.0958250	0.095825	0.09582	0.095826	0.095825
			Av.	0.0891568	0.095825	0.09582	0.095826	0.095825
			Worst	0.0291438	0.095825	N.A.	0.095826	0.095825
G9	min	680.6300573	Best	680.91	680.630	680.630	680.631592	680.63008
			Av.	681.16	680.656	680.641	680.643410	680.63642
			Worst	683.18	680.763	N.A.	680.719299	680.69832
G10	min	7049.3307	Best	7147.9	7054.316	7061.13	7051.902832	7059.86350
			Av.	8163.6	7559.192	7497.434	7253.047005	7509.32104
			Worst	9659.3	8835.655	N.A.	7638.366211	9398.64920
G11*	min	0.75	Best	0.75	0.750	0.75	0.749090	0.7499990
			Av.	0.75	0.750	0.75	0.749358	0.7499990
			Worst	0.75	0.750	N.A.	0.749830	0.7499990
G12	min	–1	Best	–0.999999857	–1.000000	N.A.	–1.000000	–1.000000
			Av.	–0.999134613	–1.000000	N.A.	–1.000000	–1.000000
			Worst	–0.991950498	–1.000000	N.A.	–1.000000	–1.000000
G13*	min	0.0539498	Best	N.A.	0.053957	N.A.	0.053986	0.0539498
			Av.	N.A.	0.057006	N.A.	0.166385	0.2977204
			Worst	N.A.	0.216915	N.A.	0.468294	0.4388511

* Problems contain equality constraints.

Table 4: Results for the welded beam design problem

Method	Best	Av.	Worst	S.D.	Av. f-evals.	Av. c-evals.
GA [5]	1.728226	1.792654	1.993408	0.074713	80,000	80,000
FSA	1.7250022	1.7564428	1.8843960	0.0424175	58,238	24,971

6.1 Welded Beam Design Problem

The welded beam design problem [5, 6] yields an optimization problem which has four design variables $x = (x_1, x_2, x_3, x_4)$ and takes the following form:

$$\begin{array}{ll} \min\limits_{x} & f(x) = 1.10471x_1^2x_2 + 0.04811x_2x_2(14 + x_2), \\ \text{s.t.} & g_1(x) = \tau(x) - 13000 \leq 0, \\ & g_2(x) = \sigma(x) - 30000 \leq 0, \\ & g_3(x) = x_1 - x_2 \leq 0, \\ & g_4(x) = 6000 - P_c(x) \leq 0, \\ & g_5(x) = \delta(x) - 0.25 \leq 0, \\ & 0.125 \leq x_1 \leq 10,\ 0.1 \leq x_2, x_2, x_2 \leq 10, \end{array}$$

where

$$\begin{array}{l} \tau(x) = \sqrt{(\tau_1(x))^2 + (\tau_2(x))^2 + \frac{x_2\tau_1(x)\tau_2(x)}{\sqrt{0.25[x_2^2+(x_1+x_2)^2]}}}, \\ \tau_1(x) = \frac{6000}{\sqrt{2}x_1x_2},\ \tau_2(x) = \frac{6000(14+0.5x_2)\sqrt{0.25[x_2^2+(x_1+x_2)^2]}}{2[0.707x_1x_2(x_2^2/12+0.25(x_1+x_2)^2)]}, \\ \sigma(x) = \frac{504000}{x_2^2x_2},\ P_c(x) = 64746.022(1 - 0.0282346x_2)x_2x_2^3,\ \delta(x) = \frac{2.1952}{x_2^3x_2}. \end{array}$$

This problem has been well studied, see [5, 6] and references therein. However, the FSA method could obtain a new solution for it which is better than the one known in the literature. Specifically, the FSA method obtained the solution

$$x^* = (0.20564426101885, 3.47257874213172, 9.03662391018928, 0.20572963979791)$$

with objective function value 1.7250022, while the known solution has the objective function value 1.728226 as reported in [5]. Moreover, the performance of the FSA method is compared against the GA-based method [5] which found the previously known solution. The best, the average, the worst and the standard deviation of objective function values obtained by 30 runs of both methods are reported in Table 4. Moreover, the average numbers Av. f-evals. and Av. c-evals. of objective and constraint functions evaluations, respectively, are also shown in Table 4. The results related to the GA-based method are taken from the original reference [5]. The figures shown in Table 4 indicate the superior performance of the FSA method.

Table 5: Results for the pressure vessel design problem

Method	Best	Av.	Worst	S.D.	Av. f-evals.	Av. c-evals.
GA [5]	6059.946341	6177.253268	6469.322010	130.929702	80,000	80,000
FSA	5868.764836	6164.585867	6804.328100	257.473670	108,883	49,253

6.2 Pressure Vessel Design Problem

The optimization problem derived from the pressure vessel design problem [5] has four design variables $x = (x_1, x_2, x_3, x_4)$. This problem can be stated as follows:

$$
\begin{aligned}
\min_{x} \; & f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \\
\text{s.t.} \; & g_1(x) = -x_1 + 0.0193x_3 \leq 0, \\
& g_2(x) = -x_2 + 0.00954x_3 \leq 0, \\
& g_3(x) = -\pi x_3^2x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \leq 0, \\
& g_4(x) = x_4 - 240 \leq 0.
\end{aligned}
$$

The FSA code was run 30 times to solve this problem and the obtained results are summarized in Table 5. The results contain the best, the average, the worst and the standard deviation of objective function values, and the average numbers of objective and constraint functions evaluations. The corresponding results of the GA-based method in Table 5 are taken from the original reference [5]. The FSA method could obtain a better solution for this problem at

$$x^* = (0.768325709391, 0.379783796302, 39.809622248187, 207.225559518596)$$

with the objective function values 5868.764836.

6.3 Tension-Compression String Problem

The problem of minimizing the weight of a tension-compression string [5] can be expressed as the following optimization problem with three design variables $x = (x_1, x_2, x_3)$:

$$
\begin{aligned}
\min_{x} \; & f(x) = x_1^2x_2(x_3 + 2), \\
\text{s.t.} \; & g_1(x) = 1 - \tfrac{x_2^3x_3}{71{,}785x_1^4} \leq 0, \\
& g_2(x) = \tfrac{4x_2^2 - x_1x_2}{12{,}566x_1^3(x_2 - x_1)} + \tfrac{1}{5{,}108x_1^2} - 1 \leq 0, \\
& g_3(x) = 1 - \tfrac{140.45x_1}{x_3x_2^2} \leq 0, \\
& g_4(x) = \tfrac{x_1 + x_2}{1.5} - 1 \leq 0.
\end{aligned}
$$

The FSA code was called 30 times with different starting solutions in order to examine the performance of the FSA method. The results obtained in all runs, as well as those of the GA-based method [5], are reported in Table 6. The results of the GA-based method are borrowed from the original reference [5]. The FSA method could obtain the better solution

$$x^* = (0.05174250340926, 0.35800478345599, 11.21390736278739)$$

Table 6: Results for the tension-compression string problem

Method	Best	Av.	Worst	S.D.	Av. f-evals.	Av. c-evals.
GA [5]	0.012681	0.012742	0.012973	0.000059	80,000	80,000
FSA	0.012665285	0.012665299	0.012665338	0.000000022	49,531	18,802

with the objective function value 0.012665285. The figures in Table 6 show that the results obtained by the FSA method are stable for this problem. Moreover, the worst solution obtained by the FSA method is still better than the best one obtained by the GA-based method [5]. Finally, the computational costs of the FSA method are much cheaper than those of the GA-based method [5].

7 Conclusion

The hybrid multi-start point-to-point FSA method is proposed. The structure of the FSA method stands on simulated annealing, filter set concept, a new solution generation procedure, and diversification and intensification schemes. These strategies are hybridized in the FSA method in such a way that a point-to-point method behaves like a population-based method without spending much computational cost. The computational results for 13 well-known test problems as well as three application problems are shown to demonstrate the efficiency of the FSA method. A superior behavior of the proposed method against population-based methods in saving the computational costs especially for the constraint function evaluations has been observed.

A List of test problems

A.1 Problem G1[2]

$$
\begin{array}{ll}
\min\limits_{x} & f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i, \\
\text{s.t.} & g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0, \\
& g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0, \\
& g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0, \\
& g_4(x) = -8x_1 + x_{10} \le 0, \\
& g_5(x) = -8x_2 + x_{11} \le 0, \\
& g_6(x) = -8x_3 + x_{12} \le 0, \\
& g_7(x) = -2x_4 - x_5 + x_{10} \le 0, \\
& g_8(x) = -2x_6 - x_7 + x_{11} \le 0, \\
& g_9(x) = -2x_8 - x_9 + x_{12} \le 0, \\
& x_i \ge 0,\ i = 1, \dots, 13, \\
& x_i \le 1,\ i = 1, \dots, 9, 13.
\end{array}
$$

[2] The formula of G1 is presented as its common form in the literature [8]. However, variable x_{13} can be eliminated since its value of the global solution, which is $x_{13} = 1$, can be easily derived.

The bounds: $U = (1, 1, 1, 1, 1, 1, 1, 1, 1, 100, 100, 100, 1)$ and $L = (0, \ldots, 0)$.
Global minimum: $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, $f(x^*) = -15$.

A.2 Problem G2

$$\begin{array}{l} \max\limits_{x} f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|, \\ \text{s.t.} \quad g_1(x) = -\prod_{i=1}^{n} x_i + 0.75 \leq 0, \\ \qquad\; g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0. \end{array}$$

The bounds: $U = (10, \ldots, 10)$ and $L = (0, \ldots, 0)$.
Best known: $f(x^*) = 0.803619$, for $n = 20$.

A.3 Problem G3

$$\begin{array}{l} \max\limits_{x} f(x) = (\sqrt{n})^n \prod_{i=1}^{n} x_i, \\ \text{s.t.} \quad h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0 \end{array}$$

The bounds: $U = (1, \ldots, 1)$ and $L = (0, \ldots, 0)$.
Global minimum: $x^* = \left(\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}} \right)$, $f(x^*) = 1$.

A.4 Problem G4

$$\begin{array}{l} \min\limits_{x} f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141, \\ \text{s.t.} \quad g_1(x) = u(x) - 92 \leq 0, \\ \qquad\; g_2(x) = -u(x) \leq 0, \\ \qquad\; g_3(x) = v(x) - 110 \leq 0, \\ \qquad\; g_4(x) = -v(x) + 90 \leq 0, \\ \qquad\; g_5(x) = w(x) - 25 \leq 0, \\ \qquad\; g_6(x) = -w(x) + 20 \leq 0, \end{array}$$

where

$$\begin{array}{rcl} u(x) & = & 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5, \\ v(x) & = & 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2, \\ w(x) & = & 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4. \end{array}$$

The bounds: $U = (102, 45, 45, 45, 45)$ and $L = (78, 33, 27, 27, 27)$.
Global minimum: $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$, $f(x^*) = -30665.539$.

A.5 Problem G5

$$\begin{aligned}
\min_x\ & f(x) = 3x_1 + 10^{-6}x_1^3 + 2x_2 + \tfrac{2}{3} \times 10^{-6}x_2^3,\\
\text{s.t.}\ & g_1(x) = x_3 - x_4 - 0.55 \le 0,\\
& g_2(x) = x_4 - x_3 - 0.55 \le 0,\\
& h_1(x) = 1000\left[\sin(-x_3 - 0.25) + \sin(-x_4 - 0.25)\right] + 894.8 - x_1 = 0,\\
& h_2(x) = 1000\left[\sin(x_3 - 0.25) + \sin(x_3 - x_4 - 0.25)\right] + 894.8 - x_2 = 0,\\
& h_3(x) = 1000\left[\sin(x_4 - 0.25) + \sin(x_4 - x_3 - 0.25)\right] + 1294.8 = 0.
\end{aligned}$$

The bounds: $U = (1200, 1200, 0.55, 0.55)$ and $L = (0, 0, -0.55, -0.55)$.
Best known: $x^* = (679.9453, 1026, 0.118876, -0.3962336)\,,\ f(x^*) = 5126.4981$.

A.6 Problem G6

$$\begin{aligned}
\min_x\ & f(x) = (x_1 - 10)^3 + (x_2 - 20)^3\,,\\
\text{s.t.}\ & g_1(x) = (x_1 - 5)^2 + (x_2 - 5)^2 + 100 \le 0,\\
& g_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 - 82.81 \le 0.
\end{aligned}$$

The bounds: $U = (100, 100)$ and $L = (13, 0)$.
Global minimum: $x^* = (14.095, 0.84296)\,,\ f(x^*) = -6961.81388$.

A.7 Problem G7

$$\begin{aligned}
\min_x\ & f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2\\
& \qquad +2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,\\
\text{s.t.}\ & g_1(x) = 4x_1 + 5x_2 - 3x_7 + 9x_8 - 105 \le 0,\\
& g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0,\\
& g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0,\\
& g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0,\\
& g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0,\\
& g_6(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0,\\
& g_7(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0,\\
& g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0.
\end{aligned}$$

The bounds: $U = (10, \ldots, 10)$ and $L = (-10, \ldots, -10)$.
Global minimum: $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$, $f(x^*) = 24.3062091$.

A.8 Problem G8

$$\begin{aligned}
\max_x\ & f(x) = \tfrac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)},\\
\text{s.t.}\ & g_1(x) = x_1^2 - x_2 + 1 \le 0,\\
& g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0.
\end{aligned}$$

The bounds: $U = (10, 10)$ and $L = (0, 0)$.
Global minimum: $x^* = (1.2279713, 4.2453733)\,,\ f(x^*) = 0.095825$.

A.9 Problem G9

$$\begin{aligned}
\min_x\ & f(x) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 \\
& \quad -10x_6 - 8x_7, \\
\text{s.t.}\ & g_1(x) = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \le 0, \\
& g_2(x) = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \le 0, \\
& g_3(x) = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \le 0, \\
& g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0.
\end{aligned}$$

The bounds: $U = (10, \ldots, 10)$ and $L = (-10, \ldots, -10)$.
Global minimum: $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$, $f(x^*) = 680.6300573$.

A.10 Problem G10

$$\begin{aligned}
\min_x\ & f(x) = x_1 + x_2 + x_3, \\
\text{s.t.}\ & g_1(x) = -1 + 0.0025(x_4 + x_6) \le 0, \\
& g_2(x) = -1 + 0.0025(-x_4 + x_5 + x_7) \le 0, \\
& g_3(x) = -1 + 0.01(-x_5 + x_8) \le 0, \\
& g_4(x) = 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0, \\
& g_5(x) = x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0, \\
& g_6(x) = x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0.
\end{aligned}$$

The bounds: $U = (10000, 10000, 10000, 1000, 1000, 1000, 1000, 1000)$ and $L = (100, 1000, 1000, 10, 10, 10, 10, 10)$.
Global minimum: $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$, $f(x^*) = 7049.3307$.

A.11 Problem G11

$$\begin{aligned}
\min_x\ & f(x) = x_1^2 + (x_2-1)^2, \\
\text{s.t.}\ & h_1(x) = x_2 - x_1^2 = 0.
\end{aligned}$$

The bounds: $U = (1, 1)$ and $L = (-1, -1)$.
Global minimum: $x^* = \left(\pm\frac{1}{\sqrt{2}}, \frac{1}{2}\right)$, $f(x^*) = 0.75$.

A.12 Problem G12

$$\begin{aligned}
\min_x\ & f(x) = 1 - 0.01[(x_1-5)^2 + (x_2-5)^2 + (x_3-5)^2], \\
\text{s.t.}\ & g_{i,j,k}(x) = (x_1-i)^2 + (x_2-j)^2 + (x_3-k)^2 - 0.0625 \le 0,
\end{aligned}$$

where $i, j, k = 1, 2, \ldots, 9$.
The bounds: $U = (10, 10, 10)$ and $L = (0, 0, 0)$.
Global minimum: $x^* = (5, 5, 5)$, $f(x^*) = 1$.

A.13 Problem G13

$$
\begin{aligned}
\min_{x} \; & f(x) = e^{x_1 x_2 x_3 x_4 x_5}, \\
\text{s.t.} \; & h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0, \\
& h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0, \\
& h_3(x) = x_1^3 + x_2^3 + 1 = 0.
\end{aligned}
$$

The bounds: $U = (2.3, 2.3, 3.2, 3.2, 3.2)$ and $L = (-2.3, -2.3, -3.2, -3.2, -3.2)$.
Global minimum: $x^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$, $f(x^*) = 0.0539498$.

References

[1] Aarts, E. and Korst, J. (2000), Selected topics in simulated annealing, in Ribeiro, C. C. and Hansen, P. (Eds.), *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers, Boston, MA.

[2] Audet, C. and Dennis Jr., J. E. (2000), A pattern search filter method for nonlinear programming without derivatives, *SIAM Journal on Optimization*, to appear.

[3] Chen, Y. X. (2001), Optimal Anytime Search for Constrained Nonlinear Programming, M.Sc. Thesis, Dept. of Computer Science, Univ. of Illinois.

[4] Coello Coello, C. A. (2002), 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, 1245–1287.

[5] Coello Coello, C. A. and Montes, E. M. (2002), Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, *Advanced Engineering Informatics* 16 , 193–203.

[6] Deb, K. (2000), An efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering* 186, 311–338.

[7] Fletcher, R. and Leyffer, S. (2002), Nonlinear programming without a penalty function, *Mathematical Programming* 91, 239–269.

[8] Floudas, C. A., Pardalos, P. M., Adjiman, C. S., Esposito, W. R., Gumus, Z., Harding, S. T., Klepeis, J. L., Meyer, C. A. and Schweiger, C. A. (Eds.) (1999), Handbook of Test Problems for Local and Global Optimization, Kluwer Academic Publishers, Boston, MA.

[9] Hamida, S. B. and Schoenauer, M. (2002), ASCHEA: New rsults using adaptive segregational constraint handling, in *Proceedings of the Congress on Evolutionary Computation (CEC2002)*, Piscataway, New Jersey, IEEE Service Center, pp. 884–889.

[10] Hedar, A. and Fukushima, M. (2002), Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization, *Optimization Methods and Software* 17, 891–912.

[11] Hedar, A. and Fukushima, M. (2003), Minimizing multimodal functions by simplex coding genetic algorithm, *Optimization Methods and Software* 18, 265–282.

[12] Hedar, A. and Fukushima, M. (2003), Tabu Search directed by direct search methods for Nonlinear Global Optimization, Technical Report 2003-007, Department of Applied Mathematics and Physics, Kyoto University.

[13] Hedar, A. and Fukushima, M. (2004), Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization, *Optimization Methods and Software*, to appear.

[14] Hock, W. and Schittkowski, K. (1981), Test Examples for Nonlinear Programming Codes, Springer-Verlag Berlin Heidelberg.

[15] Kelley, C. T. (1999), Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition, *SIAM Journal on Optimization* 10, 43–55.

[16] Kelley, C. T. (1999), Iterative Methods for Optimization, SIAM, Philadelphia, PA.

[17] Kirkpatrick, S., Gelatt Jr., C. D. and Vecchi, M. P. (1983), Optimisation by simulated annealing, *Science* 220, 671–680.

[18] Koziel, S. and Michalewicz, Z. (1999), Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, *Evolutionary Computation* 7(1), 19–44.

[19] Laarhoven, P. J. (1988), Theoretical and Computational Aspects of Simulated Annealing, Stichting Mathematisch Centrum, Amsterdam.

[20] Laarhoven, P. J. and Aarts, E. H. (1987), Simulated Annealing: Theory and Applications, D. Reidel Publishing Company, Dordrecht, Holland.

[21] Laguna, M. and Martí, R. (2002), Experimental testing of advanced scatter search designs for global optimization of multimodal functions, Working paper, University of Colorado at Boulder.

[22] Laguna, M. and Martí, R. (2003), Scatter Search: Methodology and Implementations in C, Kluwer Academic Publishers, Boston.

[23] Martí, R. (2002), Multi-start methods, in Glover, F. and Kochenberger, G. (Eds.), *Handbook of MetaHeuristics*, Kluwer Academic Publishers, Boston, MA, pp. 355–368.

[24] Martí, R. and Moreno, J.M. (2003), Métodos multi-arranque, *Inteligencia Artificial* 19, 49–60.

[25] Montes, E. M. and Coello Coello, C. A. (2003), A simple multimembered evolution strategy to solve constrained optimization problems, Technical Report EVOCINV-04-2003, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México D.F., México.

[26] Michalewicz, Z. and Schoenauer, M. (1996), Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation* 4(1), 1–32.

[27] Nelder, J. A. and Mead, R. (1965), A simplex method for function minimization, *The Computer Journal* 7, 308–313.

[28] Romeijn, H. E. and Smith, R. L. (1994), Simulated annealing for global constrained optimization, *Journal of Global Optimization* 5, 101–126.

[29] Runarsson, T. P. and Yao, X. (2000), Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation* 4(3), 284–294.

[30] Schoen, F. (2002), Two phase methods for globale optimization, in Pardalos, P. M. and Romeijn, H. E. (Eds.), *Handbook of Global Optimization*, Kluwer Academic Publishers, Boston, MA, pp. 151–178.

[31] Wah, B. W. and Chen, Y. X. (2000), Optimal anytime constrained simulated annealing for constrained global optimization, in Dechter, R. (Ed.), LNCS 1894, Springer-Verlag, pp. 425–440.

[32] Wah, B. W. and Wang, T. (2000), Tuning strategies of constrained simulated annealing for nonlinear global optimization, *International Journal on Artificial Intelligence Tools* 9(1), 3–25.

[33] Wang, T. (2000), Global Optimization of Constrained Nonlinear Programming, Ph.D. Thesis, Dept. of Computer Science, Univ. of Illinois.

[34] Wang, P. P. and Chen, D. S. (1996), Continuous optimization by a variant of simulated annealing, *Computational Optimization and Applications* 6, 59–71.