# TECHNICAL RESEARCH REPORT

## *A Population-Based Search from Genetic Algorithms through Thermodynamic Operation*

*by R-L Sun, J.E. Dayhoff and W.A. Weigand*

**T.R. 94-78**




*Sponsored by*
*the National Science Foundation*
*Engineering Research Center Program,*
*the University of Maryland,*
*Harvard University,*
*and Industry*

# A Population-Based Search from Genetic Algorithms through Thermodynamic Operation

Ray-Long Sun, Judith E. Dayhoff and William A. Weigand

University of Maryland
College Park, MD 20742

October 31, 1994



This work was supported in part by the Institute for Systems Research at the University of Maryland, under NSF Grant CDR-88-03012.

# Abstract

The guided random search techniques, genetic algorithms and simulated annealing, are very promising strategies, and both techniques are analogs from physical and biological systems. Through genetic algorithms, the simulation of evolution for the purposes of parameter optimization has generally demonstrated itself to be a robust and rapid optimization technique. The simulated annealing algorithm often finds high quality candidate solutions. Limitations, however, occur in performance because optimization may take large numbers of iterations or final parameter values may be found that there are not at global minimum (or maximum) points. In this paper we propose a population-based search algorithm that combines the approaches from genetic algorithms and simulated annealing. The combined approach, called GASA, maintains a population of individuals over a period of generations. In the GASA technique, simulated annealing is used in choices regarding a subset of individuals to undergo crossover and mutation. We show that the GASA technique performs superior to a genetic algorithm on the Bohachevsky function, an objective function with m any local minima. The methodology and the test results on function optimization are given and compared with classical genetic algorithms.

# 1 Introduction

In an optimization problem, one attempts to determine the best candidate solutions to minimize or maximize an objective function, subject to constraints dictated by the application. Typically, the inherent complexity or the uncertainty surrounding it prevents one from specifying an acceptable *a priori* solution. These complex parameter optimization problems have been approached with advanced search techniques such evolutionary computation or combinatorial methods, which are different from traditional calculus-based techniques [14].

Evolution is a remarkable problem-solving scheme. Genetic Algorithms (GAs) are an attractive class of computational models that attempt to mimic the mechanisms of natural evolution to solve problems in a wide variety of domains. The theory behind GAs was proposed by John Holland in his landmark book [9]. GAs use a direct analogy to natural events. They work with a *population* of "individuals", each representing a possible solution to a given problem. Each individual is assigned a "fitness score" according to how good a solution to the problem it is. The highly fit individuals are given opportunities to "reproduce", by "crossover" with other individuals in the population, or by "mutation" for an individual in a certain generation. These procedures produce new individuals as "offspring", which share some features taken from each "parent". The least fit members of the population are less likely to get selected for reproduction, and so they "die out".

GAs do not always converge well. GAs start with an initial random population, and allocate increasing trials to regions of the search space found to have high fitness. This is a disadvantage if the maximum is in a small region, surrounded on all sides by regions of low fitness. Another search technique based on physical, rather than biological processes, can help GAs' generations converge better, and still maintain the simplicity of the iterated search. This promising method from

metallurgy, called simulated annealing, enables technology from statistical thermodynamics to be used on combinatorial population problems. The annealing technique was invented by Kirkpatrick in 1983 [10]. It is essentially a modified version of hill-climbing. Starting from a random point in the search space, a random move is made. If the move takes us to a higher point, it is accepted. If it takes us to a lower point, it is accepted only with the probability calculated at that time. The probability begins close to 1, but gradually reduces towards zero — the analog being with the cooling of a solid. A description of the cooling of molten metals motivates this algorithm. After slow cooling (annealing), the metal arrives at a low energy state. In such circumstances, thermal equilibrium at a given temperature is characterized by a Boltzmann distribution function of the energy states. Under these conditions, even at low temperature, a transition may occur from a low to high energy level, albeit with a small probability. Such transitions are assumed to be responsible for the system reaching a minimum energy state instead of being trapped in a local meta-stable state.

Like other random search techniques, simulated annealing only deals with one candidate at a time, and so does not build up an overall picture of the search space. No information is saved from previous moves to guide the selection of new moves. To overcome this limitation, we use the population-based guide search in combination with the annealing technique to create a new search algorithm. Sirag and Weisser [18] use a similar approach with a "thermally" motivated adaptation of inversion, mutation and crossover on the TSP problem; however, our approach specifically creates subpopulations that are subject to mutation or crossover and are created by an annealing algorithm.

In the GASA technique, objective function parameters are coded into a "chromosome" and a

population of individuals is maintained and updated over a period of generations. When genetic algorithms are used alone, usually the most fit individuals are preserved for the next generation but any individuals can be selected to undergo crossover and mutation. In the GASA technique, simulated annealing is used in choices regarding the subset of individuals to undergo crossover and mutation. At each generation, some individuals are accepted for a special gene pool that contains mostly the fittest individuals. A simulated annealing approach, with a probability calculation based on a Boltzmann distribution using an energy value parameter, is used to decide which individuals are accepted or rejected. Those set aside from the special gene pool are subjected to crossover. A similar procedure is then used to decide which individuals will undergo mutations.

This work differs from the previous work of Lin *et al.*[12], which makes steps on an individual by individual basis by simulated annealing, combined with a genetic approach. We use subpopulations and apply the SA approach to the selection of which individuals are placed in which subpopulations. Only some populations undergo crossover and mutation. Other attempts have been made to improve GAs, but without including a SA approach [17, 15, 1]

The GASA approach is intuitively appealing because it allows the population to sustain the most fit individuals in most cases, but in a few cases, due to the probabilistic nature of the simulated annealing, highly fit individuals are crossed over or mutated. The less fit individuals are more often subjected to the perturbations of crossover and mutation, and these are the individuals who could benefit the most from such changes. Fewer iterations are required with GASA to arrive at the final solution at the global minimum of the objective function. Thus the combined technique appears promising for difficult problems with many local minima. Potential applications include controller design and training [20], molecular prediction [19, 21, 3], fuzzy reasoning and control [16], genetic

algorithms to neural networks [11].

In Section 2 we give a summary of the genetic algorithm approach followed by a description of simulated annealing. In Section 3, we present the unified approach that results in a class of GASA algorithms, and we show the specific algorithm used here. In Section 4 we give results on the application of GASA and GAs to a sample problem, that of finding the global minimum in the Bohachevsky function. Section 5 gives a discussion of conclusions and impact from this work.

# 2 Search Problems and Approaches

In an optimization problem, one attempts to determine the "best" candidate solutions to certain mathematically defined problems, which are often models of physical reality. Ordinarily, a typical value problem is a function optimization. An objective (cost) value is found from the applied problem, then the search for the candidate solutions will start.

Traditional search in these problems are based on hill-climbing methods which move point by point. Simulated annealing provides a point by point search with the added property that a hill can be climbed according to the point's transition probability. Genetic algorithms provide a candidate population in the domain space, which can uncover a broad population of possibilities during the search. The proposed algorithm utilizes the characteristics of population search and thermodynamic equilibrium to simulate a natural mechanism for solving mathematical problems. However, there is an extremely diverse range of practical applications.

In this section we describe the genetic algorithm approach and simulated annealing technique. Each was invented independently as a search and optimization technique designed to overcome local minima and to (hopefully) arrive at a global minimum. In genetic algorithms, a population of

individuals is maintained, and this population can be spread about parameter space to increase the likelihood of having one individual near to a global minimum. With simulated annealing, the global minimum is sought through probabilistic moves of one individual that may allow that individual to climb a hill from a local to a global minimum point. Thus, two vastly different approaches were invented with similar goals.

## 2.1 An Overview of Genetic Algorithms (GAs)

GAs work on bit strings of fixed length $l$, i.e. $U = \{0,1\}^l$. During generation $t$, a genetic algorithm maintains a population of candidate solutions (chromosomes), $\vec{P}(t) = \{\vec{\nu}_i(t) | \nu_i \in U, i = 1, 2, \cdots, n\}$, where $\vec{P}(t)$ is a set of chromosomes (bit strings) $\vec{\nu}_i(t)$ at time $t$, and $n$ is the number of individuals. Elements of $\vec{\nu}_i(t)$ are $\nu_{ik}, k = 1, 2, ..., m$, where $m$ is the length of the chromosome. In multi-dimensional function optimization, $D$ is the dimension of the function. The selection procedure probabilistically selects an individual $i$ to remain in the population and reproduce with probability

$$p_{s_i} = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{1}$$

The fitness $f_i$ is calculated from the objective function. Those states not selected are culled from the population. The average fitness of the population is defined as $\bar{f} = \sum_{j=1}^{n} f_j / n$. There are $b_i(t)$ copies of an individual $i$ at time $t$, and the new population will have $b_i(t+1) = b_i(t) f_i / \bar{f}$ copies of individual $i$ at time $t+1$. The effect of this reproduction scheme is that above average performing individuals reproduce more, replacing poorly performing individuals.

Each offspring solution $x_i^t$ is evaluated to give some measure of its "fitness". Then a new population at generation $t+1$ is formed to generally have more fit individuals. Some members of the new population undergo reproduction (replication) by means of crossover and mutation, to

form the new solutions. Crossover combines the features of two parent chromosomes to similar offspring by swapping corresponding segments of the two parents. As an example, consider of two six-dimensional vectors:

$$A = a_1\, a_2\, a_3 | a_4\, a_5\, a_6$$

$$B = b_1\, b_2\, b_3 | b_4\, b_5\, b_6$$

For crossover position at $k = 3$, the offspring after resulting crossover would be:

$$A' = a_1 a_2 a_3 b_4 b_5 b_6$$

$$B' = b_1 b_2 b_3 a_4 a_5 a_6$$

The intuition behind the application of crossover results in a randomized yet structured information exchange. Each solution created combines the characteristics of both parents.

Mutation arbitrarily alters one or more genes of a selected individual, by a random change with a probability equal to the mutation rate. The intuition behind the mutation operator is the introduction of new variability into the population.

## 2.2 Simulated Annealing and Thermodynamic Equilibrium

Statistical mechanics is the basis for studying the behavior of annealing, such as atoms (data points) in a fluid (a data set), in thermal equilibrium at a finite temperature. Suppose that the state of the system is identical with the set of spatial positions of the components. Here, the phase transition is defined as data transform between *ordered* (low energy) and *disordered* (high energy) phases. The *ordered phases* obey certain types of data arrangements. If the system is in thermal equilibrium at a given temperature $T$, then the probability of a given state $s$ depends upon the energy $E(s)$ of the

state and follows the Boltzmann distribution:

$$p_T(s) = \frac{e^{\frac{-E(s)}{KT}}}{\sum_{\omega \in S} e^{\frac{-E(\omega)}{kT}}} \tag{2}$$

where $k$ is the Boltzmann constant and $S$ is the set of all possible states.

**Metropolis Acceptance Criterion** One can simulate the behavior of a system of particles in thermal equilibrium at temperature $T$ by using a stochastic relaxation technique developed by Metropolis *et al* (1953) [13]. Suppose that at time measure $t$, the system is in state $\mathbf{q}$. A candidate $r$ for the state at time $t+1$ is generated randomly. The criterion for selecting or rejecting state $\mathbf{r}$ depends on the difference between the energies of state $\mathbf{r}$ and $\mathbf{q}$. Specifically, one computes the ratio $h(\triangle E)/T$ ($T$ replaces $kT$), and $\triangle E = (E(\mathbf{r}) - E(\mathbf{q}))$, where:

$$h = \frac{p_T(\mathbf{r})}{p_T(\mathbf{q})} = e^{\frac{-\triangle E}{T}} \tag{3}$$

If $h > 1 (\triangle E < 0)$, that is, the energy of $\mathbf{r}$ is strictly less than the energy of $\mathbf{q}$, then the state is automatically accepted as the new state for time $t+1$. If $h \leq 1$, that is, $E(\mathbf{r}) \geq E(\mathbf{q})$, then the state $\mathbf{r}$ is accepted as the new state with probability $h$. Thus, states of higher energy can be attained. It can be shown that as $t \rightarrow \infty$, the probability that the system is in a given state $s$ equals $p_T(s)$, regardless of starting state, and thus the distribution of states generated converges to the Boltzmann distribution [5]. Theoretically, if the result is a state of low energy, then the change attempted with high probability. The probability that a transition occurs with an increase in energy level decreases with the temperature, thus reflecting the higher disorder occurring at high temperatures. However, when equilibrium is established, the accessible state with low energy is more likely to occur.

Kirkpatrick's annealing algorithm [10] consists of running the Metropolis Monte Carlo integration algorithm at each temperature in the annealing schedule for the amount of time prescribed

by the schedule, and selecting the final state generated as a near-optimal solution. The Metropolis algorithm, which accepts states that increase cost as well as those that decrease cost, is the mechanism for avoiding entrapment at a local minimum.

The annealing process is inherently slow. Geman and Geman (1984) [5] determine an annealing schedule for convergence to a minimum with sufficient temperature decreasements. Specifically, for a given sequence of temperatures $\{T_t\}$ such that $T_t \to 0$ as $t \to \infty$ and $T_t \geq \frac{T_0}{logt}$ for a large constant $T_0$ (the initial temperature), then the probability that the system is at state $s$ as $t \to \infty$ is equal to $p_0(s)$. Thus, given enough time simulated annealing will yield a global optimum almost with certainty even when started in an arbitrary state. The more slowly annealing is applied by decreasing temperature, the higher probability that the system attains the global minimum.

In a finite-time behavior of simulated annealing, the Markov chain corresponding to each temperature is applied for only a finite number of steps. This finite-time behavior of simulated annealing is determined by the cooling schedule and by the generating chain. Therefore, the analysis of the finite-time behavior depends on both of these components of the algorithm.

# 3 A Unified Approach

## 3.1 GASA Algorithm Design

Before we describe how to design Genetic Algorithms + Simulated Annealing (GASA), let's formulate a general statement of searching optimum problem. Consider a general function minimization problem, with function $f : \vec{X} \to \mathcal{R}^{\mathcal{D}}$, $\vec{X}$ is the set of all variables,

$$\begin{aligned} J &= \min\ f(\vec{X}) \\ \vec{X} &= \{x_i, x_2, \cdots, x_D\} \end{aligned} \tag{4}$$

$$x_i \in [u_i, v_i]$$

where $u_i$ and $v_i$ are the lower and upper bounds, respectively, and $D$ is the number of the variables $x_i$ in the defined domain of the function $f$. When we apply the generation mechanism of GAs, the $x_i$ is replaced by $x_i(t)$, where $t$ is the number of generations. Our goal for this problem is to search for the global minimum $f^*$, where $x_i^*$ is the minimum location.

1. Temperature schedule

   Simulated annealing offers a strategy very similar to iterative improvement with one major difference: annealing allows perturbations to move in a controlled fashion. We now refer to individual perturbations as *moves*. Each move can transform one configuration into an either *better* or *worse* configuration, so it is possible to jump out of a local minima and potentially fall into a more promising path. The moves are controlled by temperature. Thus, the basic requirement for simulating this process is the ability to simulate how the system reaches thermodynamic schedule at each temperature. However, for the temperature schedule, the initial temperature should be set large enough to be successful in the annealing process. We fix the annealing schedule as $T_{m+1} = \alpha \cdot T_m$, where $\alpha$ is a temperature reduction parameter usually assigned to a value between 0.99 and 0.85, and $m$ is the number of moves of individuals attempted to reach equilibrium.

2. Population generating

   First, we have to create an initial population of individuals, where each individual is a binary vector of $l$ bits. The initial population with $D$ individuals is noted as $\vec{P}(0) = \{\vec{v}_i(0) | i = 1, 2, \cdots, D\}$. All of the $l$ bits for each individual are initialized. The creating mode is broadly applied in the following operation. The vector, $\vec{v}(0)$ is the initial encoded Boolean binary

population vector which can be decoded into genes $x_i^t$. The initial population is prescribed at $t = 0$. It can be seen as an encoded population matrix $\vec{P}(t)$ at generation $t$ and can be formulated as

$$\vec{P}(t) = \{\vec{v}_i(t)\} \triangleq \left( \begin{array}{c|c|c|c} \overbrace{v_{111}v_{112}\cdots v_{11l}}^{x_{11}} & \overbrace{\cdots}^{x_{12}} & \cdots & \overbrace{v_{1D1}\cdots v_{1Dl}}^{x_{1D}} \\ \overbrace{v_{211}v_{212}\cdots v_{21l}}^{x_{21}} & \overbrace{\cdots}^{x_{22}} & \cdots & \overbrace{v_{2D1}\cdots v_{2Dl}}^{x_{2D}} \\ \cdots\cdots & \cdots & \cdots & \cdots\cdots \\ \overbrace{v_{n11}v_{n12}\cdots v_{n1l}}^{x_{n1}} & \overbrace{\cdots}^{x_{n2}} & \cdots & \overbrace{v_{nD1}\cdots v_{nDl}}^{x_{nD}} \end{array} \right)_t \tag{5}$$

where $\{n, D, l\}$ are the total number of individuals, dimension (the number of chromosomes per individual), and bits (the number of genes per chromosome), respectively. The vector set, $\{\vec{v}_i(t)\}$, is the set of all binary values of the parameters of individual $i$ in a population. For convenience, we can define a binary matrix $\underline{\underline{v}}$ as

$$\underline{\underline{v}}(t) \stackrel{\text{def}}{=} \{v_{ijk}(t) | i = 1, \ldots, n, j = 1, \ldots, D, k = 1, \cdots, l\}$$

Its decoded real-value population matrix $\underline{\underline{X}}(t)$ is obtained by applying the decoding operator $\Gamma$

$$\Gamma\{\vec{v}(t)\} = \underline{\underline{X}}(t) \triangleq \left( \begin{array}{cccc} x_{11} & x_{12} & \ldots & x_{1D} \\ x_{21} & x_{22} & \ldots & x_{2D} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \ldots & x_{nD} \end{array} \right)_t \tag{6}$$

where $n$ is the total number of individuals and $\Gamma$ is a decoding function. Each line notates the parameters of an individual.

3. Function evaluation

As noted in GAs, the functional evaluation plays an important role, rating potential candidate solutions in terms of their objective function value, which ultimately determines fitness. The individuals need to be decoded because of the binary bits. The function evaluation mode can

be defined as

$$eval(\underline{\underline{X}}(t)) \stackrel{\text{def}}{=} f(\{\vec{X}_i(t)\}) \triangleq \begin{pmatrix} f(\vec{X}_1(t)) \\ f(\vec{X}_2(t)) \\ \vdots \\ f(\vec{X}_n(t)) \end{pmatrix} \tag{7}$$

where $n$ is the number of the individuals in the population at generation $t$. The vector set, $\{\vec{X}_i(t)\}$, is the real-value of individual $i$ at generation $t$. Then it still requires the scaling of fitness from the evaluated function values.

4. Fitness scaling

The scaling function $\delta$ calculates the fitness from the objective function and it scales the fitness. The function $\delta$ is able to assure the best individual receives the largest fitness. Normally, fitness evaluation transforms the cost function value $f$ into a measure of relative fitness. Most commonly, a *linear dynamic scaling* is used which takes into account the worst individual of the population $\vec{P}(t-\omega)$ over $\omega$ time steps before (if $t-\omega < 0$ then replace $\vec{P}(t-\omega)$ with $\vec{P}(0)$). If $\omega$ is a scaling window, the scaling function $\delta$ is defined by

$$\delta(f(\Gamma(\underline{\underline{\nu}}(t))), \omega) = a \cdot f(\Gamma(\underline{\underline{\nu}}(t))) + b(\omega) \tag{8}$$

where $t$ is the current generation, $a$ is a scaling factor ($a = -1$ usually). The $b(w)$ is a baseline reflecting the worst fitness value which occurred within the last $\omega$ generations. Each individual's fitness is subtracted from $b(\omega)$. With linear dynamic scaling, negative fitnesses do not occur and thus do not cause any problem for the calculation of selection probabilities according to Equation (1).

5. Selection

The selection procedure probabilistically selects an individual $i$ to remain in the population and reproduce with probability

$$p_i = \frac{\Phi_i}{\sum_{j=1}^{n} \Phi_j} \tag{9}$$

The fitness $\Phi_i$ is calculated from the objective function and defined as below.

$$\Phi(\underline{\underline{\nu}}) \stackrel{\text{def}}{=} \delta(f(\Gamma(\underline{\underline{\nu}})), \omega) \tag{10}$$

where $\Phi$ is a fitness conversion operator. When the individuals are selected, the high-fitness individuals will be reproduced according to their fitness function $\Phi$.

6. Genetic Alterations through Thermodynamic Operation

Before the selected individuals process crossover, the modified Metropolis criterion is applied to decide which individuals are to be put into a subpopulation that does crossover. This decision is made according to an acceptance probability. First, choose the minimum point of the individuals as the comparison center. The minimum objective function value and the optimum location (selection center) are given by $\{f^*, \vec{X}^*\}$ for the current population.

It is possible to define each individual's energy function from the objective function value. Thermodynamics indicates that thermal equilibrium at temperature $T$ is a probability distribution in which a state with *energy function* $\{E_i(t) = f(\vec{X}_i(t)) | i = 1, \cdots, n\}$, whose $n$ is the total number of individuals in generation $t$. The corresponding *threshold probability* $p_i$ is

$$\frac{e^{-\frac{E_i}{kT}}}{Z(T)} \tag{11}$$

where $Z(T)$ is a partition factor.

From the above definition, the probability $p$ between $\vec{X}_i$ and $\vec{X}_j$ is

$$\begin{aligned} \frac{p_i}{p_j} &= \frac{e^{-\frac{E_i}{kT}}}{e^{-\frac{E_j}{kT}}} \\ &= e^{-\frac{\triangle E}{kT}} \end{aligned} \tag{12}$$

where $\triangle E = E_i - E_j$. The $k$ is 1 in this simulation.

**Modified Metropolis Criterion (MMC)**

A modified Metropolis criterion is used here. The method applied here is to replace $E_j$ with the population's minimum energy $E^*$, and to have $\vec{X}_j$ replaced by $\vec{X}^*$. This modification results in $\triangle E$ being greater than or equal to 0. Thus, in classic simulated annealing, sometimes the uphill climbing is retained and the move *may* still occur. Therefore, a new, higher energy state will be accepted by the individual, according to a threshold probability. In GASA, an entire population of individuals is divided into two classes, say $\Xi_c$ & $\Xi_{nc}$. Usually, the motivation of division is from the individual's energy (expressed with its threshold probability). For example, the crossover operation increases the diversities in the low-energy class $\Xi_c$. Occasionally because of the SA aspect, the higher-energy individual may move uphill or downhill according to their crossover mating. When crossover is done, the resulting individuals may end up with a higher energy level, that allows them to climb over a hill and into another basin. Since the new basin may have a lower minimum, there can be beneficial to let the individuals reach the valley.

For each individual $i$, $p_i$ is calculated. If $p_i > random[0, 1)$, then the individual is accepted and put in a subpopulation pool $\Xi_c$. Otherwise the individual is not accepted into $\Xi_c$ and is put into subpopulation $\Xi_{nc}$. We process the set of unaccepted individuals by the crossover operator $C_{\{p_c\}}$, where $p_c$ is the applied crossover rate, on population $\Xi_{nc}$. The mates of

individuals are crossed-over by selecting a cross-point randomly. The crossover cannot always guarantee the offspring's fitness would be higher than that of their parents, but it can increase diversity and sometimes makes recombinations that have superior fitness. The typical feature of this algorithm is that, besides accepting individuals with high fitness in population $\Xi$ (after selection), it to a large extent puts individuals with high fitness into population $\Xi_c$ and puts individuals with low fitness into population $\Xi_{nc}$, according to the temperature. Initially, at large values of $T$, exceptions are more likely; as $T$ decreases only smaller exceptions will be made to the general rule that high fitness individuals are preserved in population $\Xi_c$.

The mutation operation will process with similar action as that in crossover. It is processed by the mutation operator $M_{\{p_m\}}$, where $p_m$ is the applied mutation rate. For each individual, we calculate $q_i$. If the individual passes the acceptance criterion, it will be kept in a gene pool $\Xi_m$. The remaining individuals are put into subpopulation $\Xi_{nm}$, then are mutated at a randomly selected bit, and kept. Thus, mutation is usually applied to the inferior genes of the running chromosomes.

By successively lowering the temperature and running this algorithm, we can simulate the individuals coming into equilibrium at each newly reduced temperature, and thus effectively find the solution candidates according to the energy transitions.

In this technique, *equilibrium* is only a conceptual criterion instead of a real physical phenomenon. It can be described as *while the generation process is repeated, a sequence of temperatory candidate solutions is produced until the solution space occupancy is described by the Equations* 11 & 12.

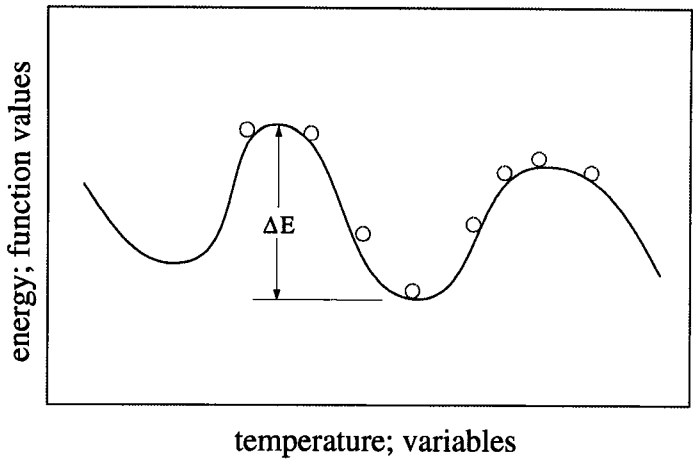


**Figure 1**: *The behavior of the participating individuals*

7. Insert the individuals in next generation

   The individuals kept in the gene pools are combined into a single pool and become the parents of the next generation $t + 1$. The initial temperature $T_0$ is chosen together with the procedures of thermal equilibrium. Then $T$ is updated and held constant, and the parameters that simulate the thermal dynamics are set. The choice of these parameters is referred to as a *cooling schedule* as recommended in step 1.

The moving of the participating individuals vs. temperature is shown in Figure 1. The balls are the individuals climbing upwards and downwards according to the temperature change. Figure 1 also reveals that the individuals' function values change with the variables.

This GASA algorithm can be described as follows:

*Parameters declaration:*
$t = 0$; *(the initial generation);*
$t_{max}$ *is the allowed maximal number of generations;*
$T_0$ = *initial temperature;*
*Tm = number of temperature moves to attempt;*

*C is the crossover operator;*

*M is the mutation operator;*

$\Xi$ *is the total gene pool per generation;*

*Initial* $\vec{P}(0) = \{\underline{\underline{\nu}}(0)\} \in U$ *where* $U = \{0,1\}^l$;

*Evaluate* $\vec{P}(0) = \{\Phi(\nu_1(0)), \cdots, \Phi(\nu_l(0))\}$;

*where* $\Phi(\nu_k(0)) = \delta(f(\Gamma(\nu_k(0))), P(0))$;

*For m =1,Tm*

*while* $(t < t_{\max})$

*For each individual i,*

*Evaluate the change of energy* $(E_i - E^*)$ *and find a probability* $p_i$;

*if threshold probability* $p_i >$ *random* $[0,1)$,

*keep the accepted individual in a gene pool* $\Xi_c$;

*else*

*put i into* $\Xi_{nc}$;

*Crossover:* $\nu_k'(t) = C_{\{p_c\}}(\vec{P}_c(t)), \forall k \in \{1, \cdots, l\}$

$, \forall \vec{P}_c(t) \in \Xi_{nc} (= \Xi - \Xi_c)$ , $p_c$ *is the crossover rate;*

*replace* $\Xi$ *with* $\Xi_c + \Xi_{nc}$;

*For each individual i,*

*Evaluate the change of energy* $(E_i - E^*)$ *of the individuals from* $\Xi$ *and* $\nu_k'(t)$,

*and find a probability* $p_i$;

*if accepted probability* $p_i >$ *random* $[0,1)$,

*keep the accepted individual in a gene pool* $\Xi_m$;

*else*

*put i into* $\Xi_{nm}$;

*Mutate:* $\nu_k''(t) = M_{\{p_m\}}(\vec{P}_m(t)), \forall k \in \{1, \cdots, l\}$

$, \forall \vec{P}_m(t) \in \Xi_{nm} (= \Xi - \Xi_m)$ , $p_m$ *is mutation rate;*

*replace* $\Xi$ *with* $\Xi_m + \Xi_{nm}$;

*Until equilibrium;*

$T = update(T)$;

*Evaluate* $\vec{P}(t) = \{\Phi(\nu_1(t), \cdots, \Phi(\nu_l(t))\}$,

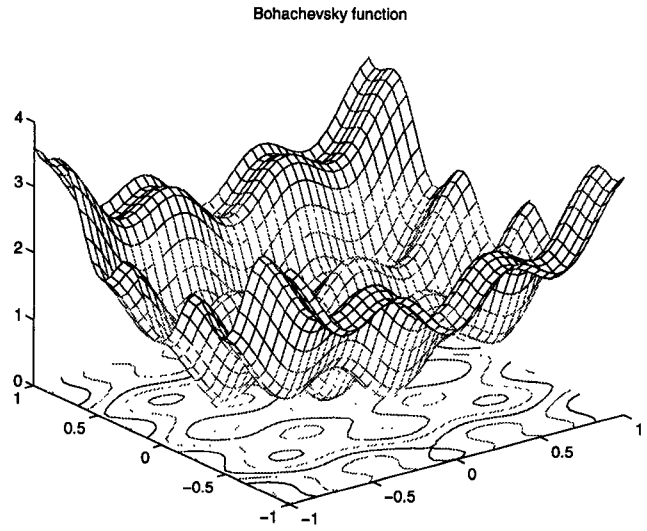


**Figure 2**: *A multiple-optima function*

*where* $\Phi(\nu_k(t) = \delta(f(\Gamma(\nu_k(t))), P(t-\omega))$;

*Select* $\vec{P}(t+1) = sel(P(t))$;

*Reproduce* $\vec{P}(t+1)$;

}; *end t*

$t = t+1$;

}; *end move*

# 4 Results on Function with Multiple Optima

To compare the GASA approach with GAs, we performed an experiment on a function with multiple optima. The function chosen is shown in Figure 2, and is generated by the following equation.

$$f(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7 \quad (13)$$

This function was examined by Bohachevsky *et al.* (1986)[2]. It has numerous local minima and a global minimum at the origin. This function is illustrated in Figure 2 with $x_1, x_2 \in [-1, 1]$. There

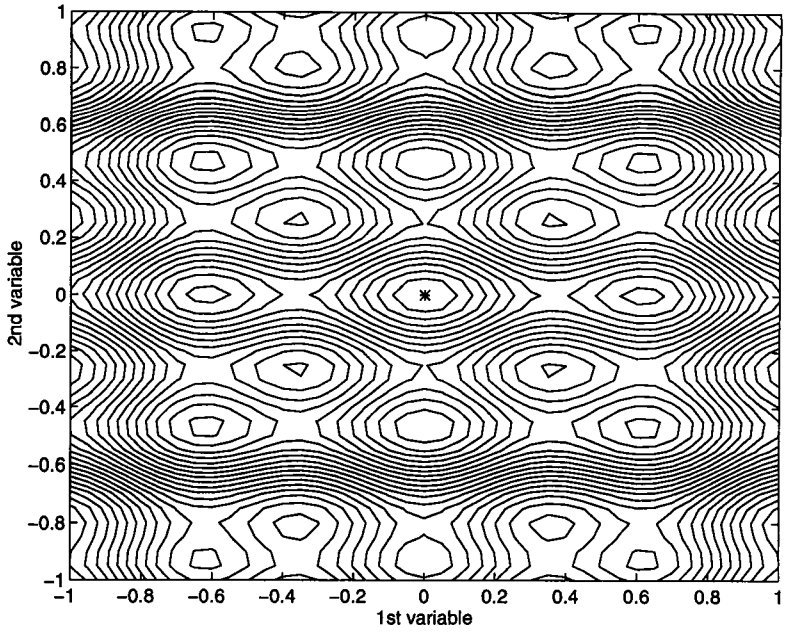


**Figure 3**: *The contour plot of Bohachevsky function*

are many local minima indicated in the contour plot as in Figure 3. "*" shows the location of global minimum. When $(x_1, x_2)$ is far from the origin, the quadratic terms of $f(x_1, x_2)$ dominate the cosine terms and the overall shape of $f$ is quadratic. If $(x_1, x_2)$ is close to the origin, the cosine functions dominate the quadratic terms and $f(x_1, x_2)$ displays many hills and valleys. It is very difficult to discover the global minimum for the function using a gradient technique.

We have applied the GASA technique to find the global optimum of the Bohachevsky function. First, we initialize the parent population randomly, and then encode the values $x_1$ and $x_2$. Secondly, using the decoded variables, we select the most fit individuals from their function values. Then, we apply the MMC (Modified Metropolis Criterion) to choose the individuals either crossed-over/mutated or kept in a gene pool. Thus, very probable moves can be rejected, and very improbable moves can be accepted – at least occasionally. However, while lowering the temperature successively, we can simulate the individuals until thermodynamic equilibrium is achieved in each generation.

Figure 4 is the annealing schedule in this experiment. The starting temperature is 4.5 and, over 28 iterations, the temperature relaxed to zero. This number of generations was sufficient to find the global minimum for this function.

Figure 5(a) shows the locations of the individuals in the 28th generation. There are two variables on each chromosomes, plotted on the horizontal and vertical axes. The objective values of all individuals in generation 28 can be found in Figure 5(b).

The best and mean score of objective values are shown in Figure 5(c) as functions of the generations. The best and mean objective values are reaching the global minimum at generation 8 and 12, respectively. The best objective values begins just below 0.3 and the mean objective value begins at about 1.4. Then there is initially a wide spread in the objective values for different individuals. After 15 generations have gone by, the mean objective value is under 0.0058, close to the global minimum of zero.

Figure 5(d) shows the value of the first and second variables over the 28 generations. These values are taken from the individual that ultimately has the best objective function. After generation 25, both variables have reached approximately 0, the location for the global minimum.

Figure 6 shows the three results from GASA, for comparison. In Figure 6(a), the fitness of the best individual is plotted over a period of generations. The highest fitness value 2 corresponds to the objective function minimum at 0. Figure 6(b) shows the standard deviation of the objective values over the entire population as a function of the generation. The min & mean of the objective values as shown as a number of function calls for this algorithm in Figure 6(c).

Figure 7 and Figure 8 are generated with GAs by applying the same crossover and mutation rate and same initial population size as those used in GASA. The same plots appear in Figures 7 & 8 for GAs as in Figures 5 & 6 for GASA. GAs become convergent after the 69th generation (see Figures 7(c) & (d)). The fittest individuals can also be found at different generations from Figure 8(a). Moreover, Figure 8(b) indicates GAs's standard deviation is an irregular oscillation. GAs use more function calls than GASA (Figure 8(c)). Thus GAs need more generations to reach convergence compared with GASA.

The comparisons of GASA and GAs with the same parameters can be found in Figures 9$\sim$12. The objective value of the mean individual in GASA decreases faster than in GAs, as shown in Figure 9. Figure 10 shows that GASA has a worse min objective value in the beginning, but sooner GASA reaches a faster convergence compared with GAs. These results illustrate that GASA can achieve better global convergence than GAs. Figure 11 shows the standard deviation of the population versus the generation, and Figure 12 shows the variance versus generation. These

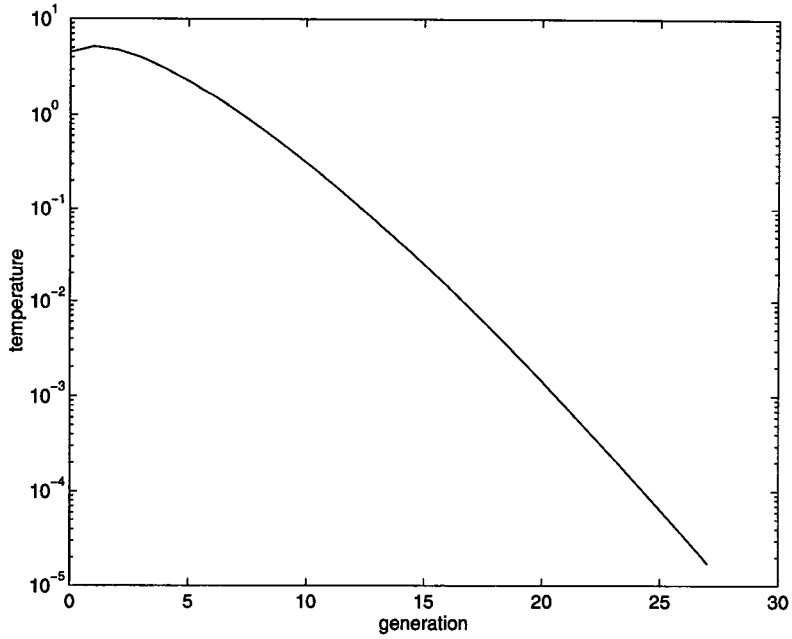


**Figure 4**: *Annealing schedule*

figures reflect decrease in population variance for GASA as the GASA method converges.

This experiment was run on SUN Sparc 10 workstation. The computation time of GASA and GAs are 32.90602 and 19.03456 CPU time, respectively. The time for GASA was longer because of annealing in each generation's population.

We have also compared twenty different runs of GASA and GAs , to show variations between runs. Twenty independent runs of GASA can be found in four rounds in Figures 13$\sim$16. The other twenty runs in four rounds of GAs are shown in Figures 17$\sim$20. From these graphs, it can be seen that it is much better for the GASA method to approach the global minimum.

In order to explore the performance of GASA and GAs, we list the compared results of the cutoff generation for differing criteria in Table I. The criterion applied depends on the precision requirement of the applications problems. In general, higher precision needs more generations, as can be seen in Table I. Column 1 shows the cutoff value of the object function, Column 2 shows the average generation at which cutoff occurred, Column 3 shows the standard deviation of the generation at which cutoff occurred, Column 4 shows the standard deviation of the objective value for the population, Column 2$\sim$4 apply to GAs, Column 5$\sim$7 show the same calculation for GASA.

**Table I**: *Bochachevsky Function: Statistic Analysis of the Number of Cutoff Gener ations to Achieve a Degree of Global Optimization with GAs*

| | GAs | | | GASA | | |
|---|---|---|---|---|---|---|
| | generation | | obj. value | generation | | obj. value |
| cutoff | mean | st. dev. | st. dev. | mean | st. dev. | st. dev. |
| $\leq 10^{-1}$ | 3.45 | 4.55 | 0.5144 | 3.80 | 5.20 | 0.3130 |
| $\leq 10^{-2}$ | 11.05 | 13.95 | 0.4933 | 8.40 | 7.60 | 0.2802 |
| $\leq 10^{-3}$ | 19.05 | 12.95 | 0.4251 | 14.60 | 5.40 | 0.1579 |
| $\leq 10^{-4}$ | 29.70 | 19.30 | 0.4030 | 18.80 | 8.20 | 0.1176 |
| $\leq 10^{-5}$ | 40.80 | 16.20 | 0.3867 | 24.10 | 2.90 | 0.0884 |
| $\leq 10^{-6}$ | 49.30 | 16.70 | 0.3806 | 29.30 | 3.70 | 0.0722 |

Twenty runs were asked for each tehnique. For example, at $10^{-4}$ cutoff, on average GASA and GAs spend 18.80 and 29.7 generations, respectively. At cutoff of $10^{-2}$ and below, GASA spends less generations than GAs to achieve the global minimum.

The standard deviations in the number of generations, shown in column 3 & 6 in Table I, for GAs are higher than those for GASA. For example, when cutoff is $10^{-3}$, for the objective value, GASA standard deviation is 0.1579 and GAs standard deviation is 0.4251. As to the average generation at cutoff $10^{-3}$, GAs' and GASA's are 19.05 and 14.60, respectively. Thus, obviously, the search with GAs is in high oscillation.

Moreover, GASA keeps less deviation in all cutoff generations, which means the operation of GASA is more stable. GAs' operation shows high oscillation, as in Figure 8(b), thus both the mean and standard deviation of GAs is larger than that of GASA. The fast decreasing of standard deviation shows how annealing is helpful to the genetic operations in GASA. The annealing is helpful to stabilize the GASA algorithm.

# 5 Conclusion

Thermodynamic operation provides a global convergence route for genetic algorithms through the GASA technique, which combines GAs and SA. Reaching the population's expected equilibrium can be aided by annealing. The performance of the population-based search for the global minimum is investigated on a sample problem with many local minima. The combination of GAs and SA

is a faster and more precise search method for optimization problems than the pure GAs on our test problem. This method is a promising approach on optimization and can provide an effective combination of principles inspired by biological and physical systems. GASA keeps better performance compared with classical genetic algorithms, from our test results. GASA is able to reach the global optimum faster, and is recommended for problems with complex objective functions and with many local minima, especially when higher precision is needed.

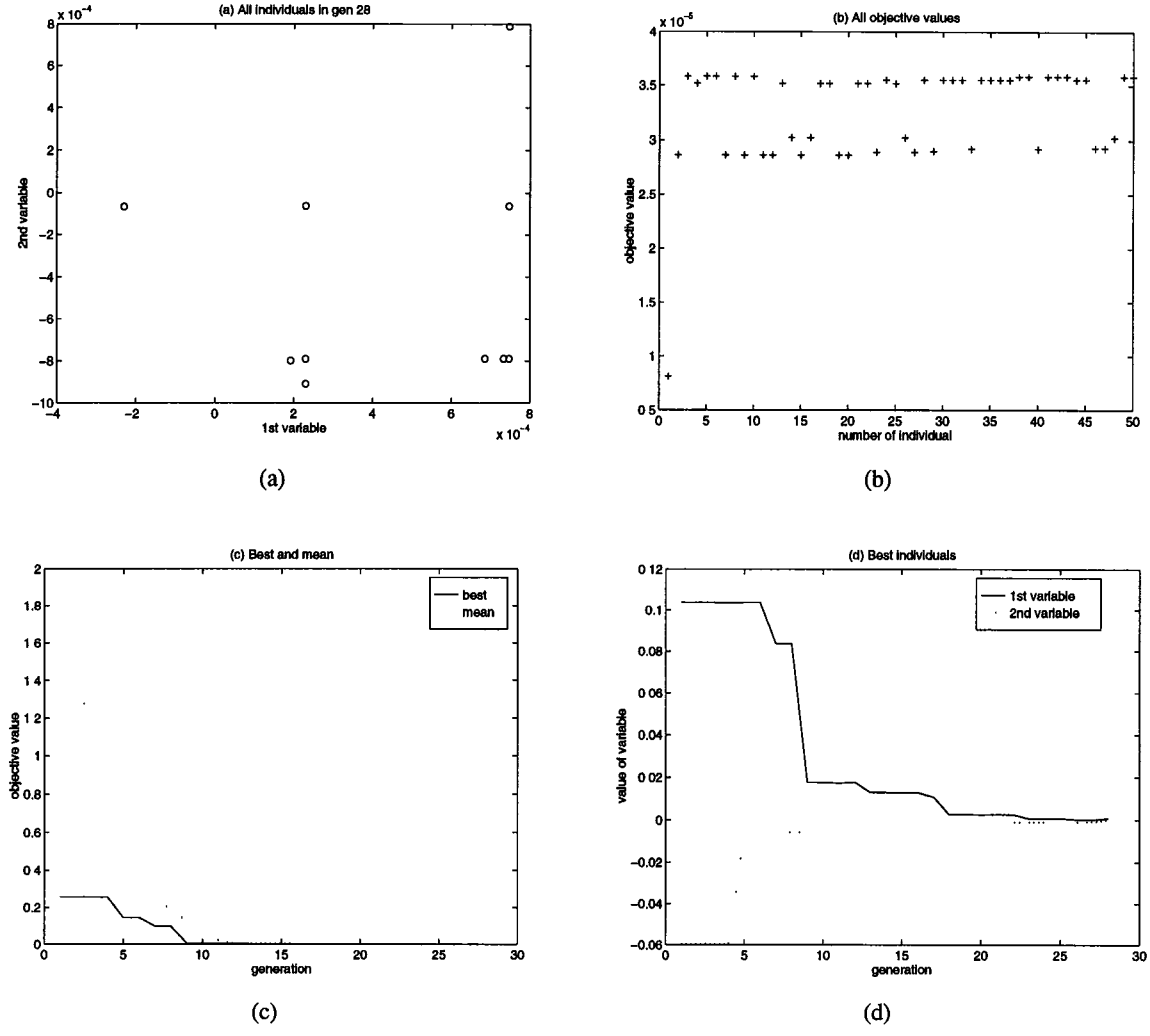


**Figure 5**: *An Interpretation of GASA operation in generations: (a) The locations of all individuals in generation 28. (b) The objective values of all individuals in generation 28. (c) Best and mean objective values of individuals in generations. (d) Best individuals in generations.*

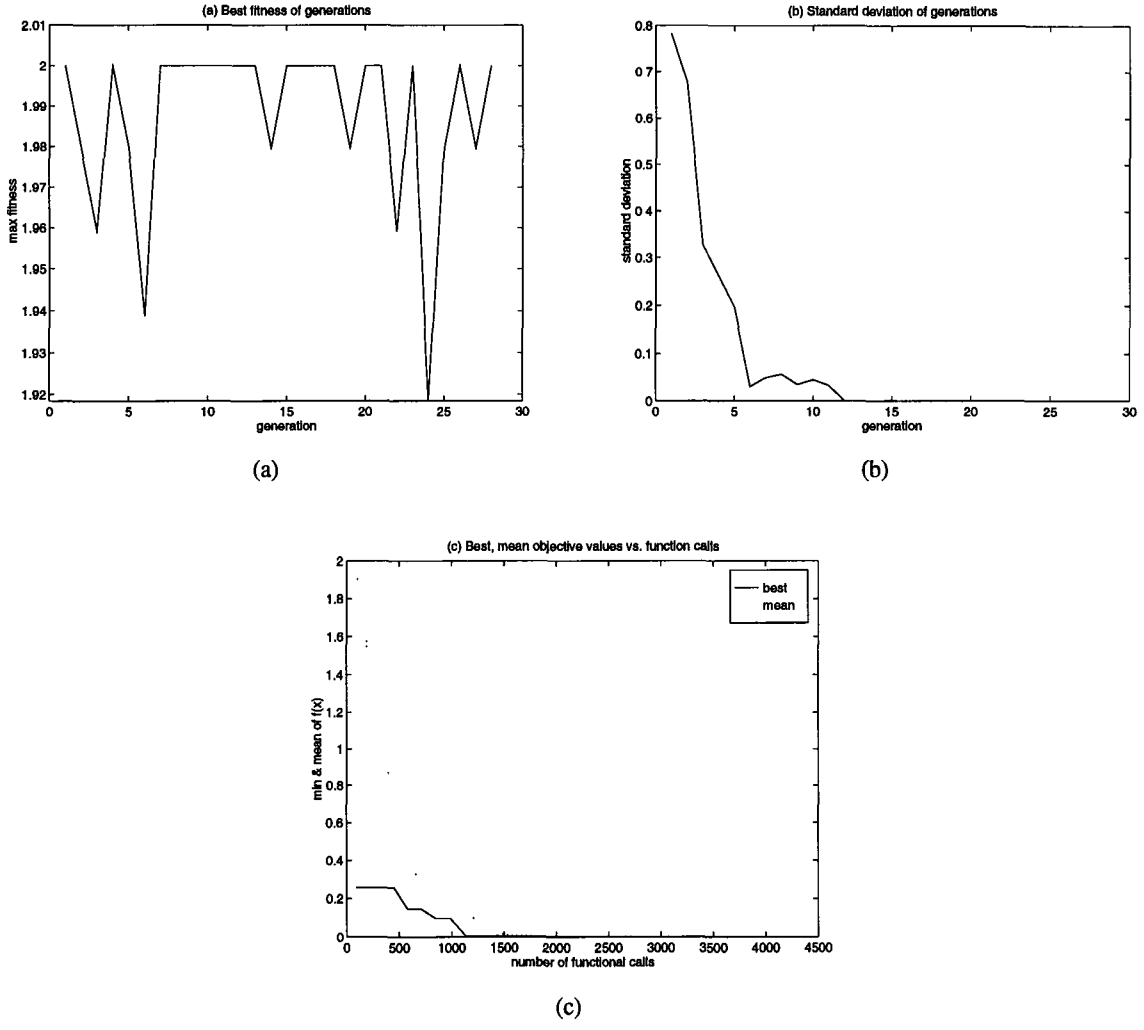


**Figure 6**: *Three comparison results of GASA: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls.*

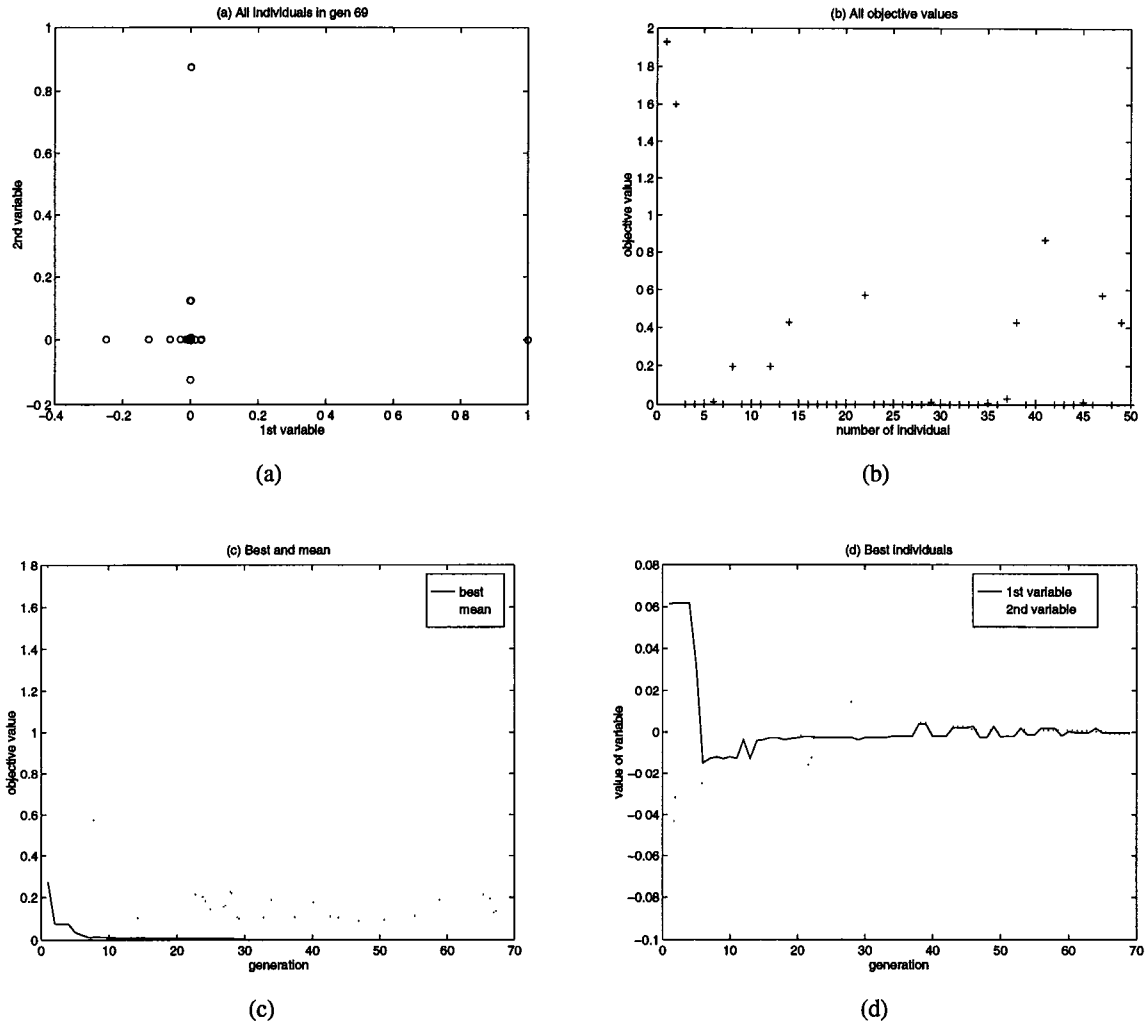


**Figure 7**: *An Interpretation of GAs operation in generations: (a) The locations of all individuals in generation 69. (b) The objective values of all individuals in generation 69. (c) Best and mean objective values of individuals in generations. (d) Best individuals in generations.*

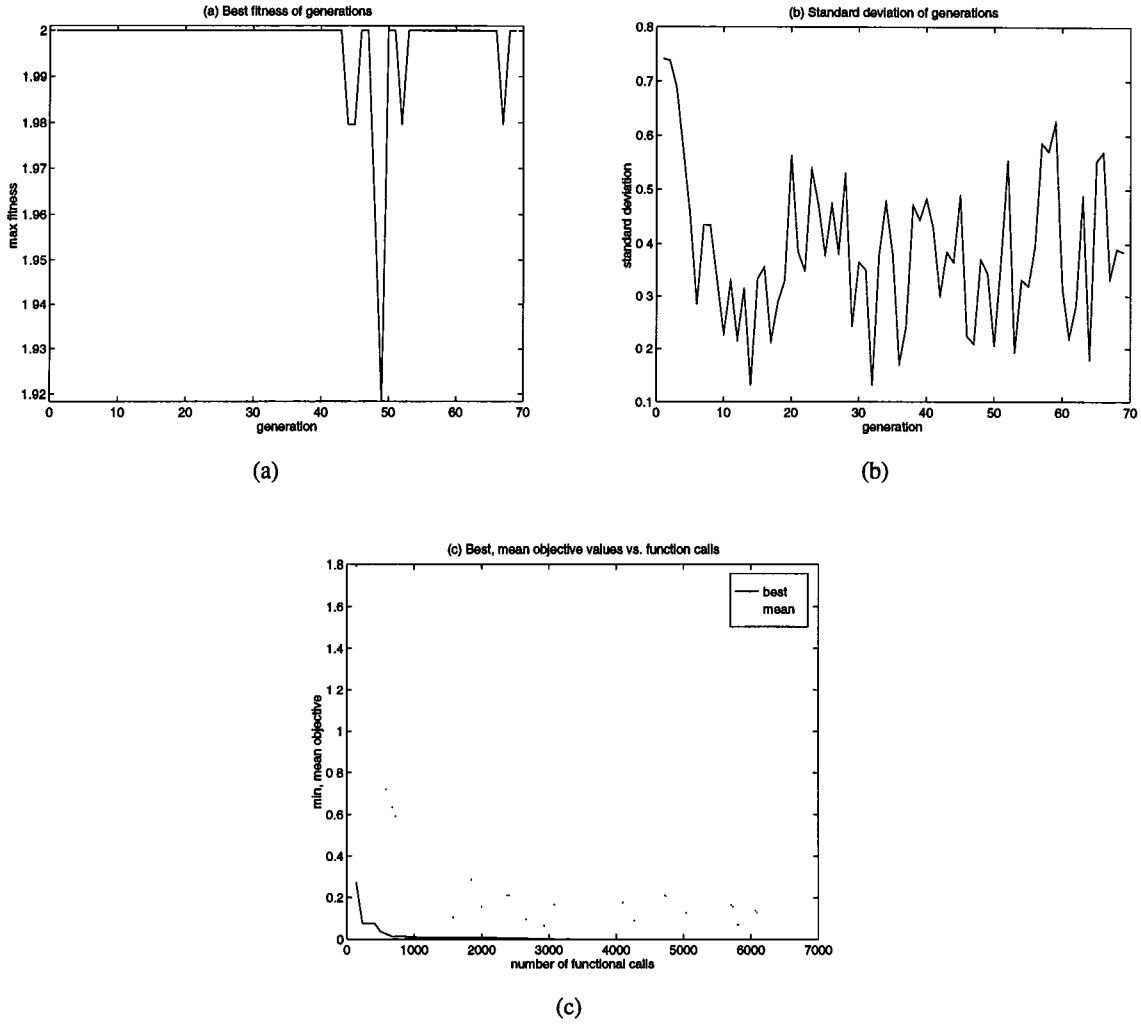


**Figure 8**: *Three comparison results of GAs: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls.*

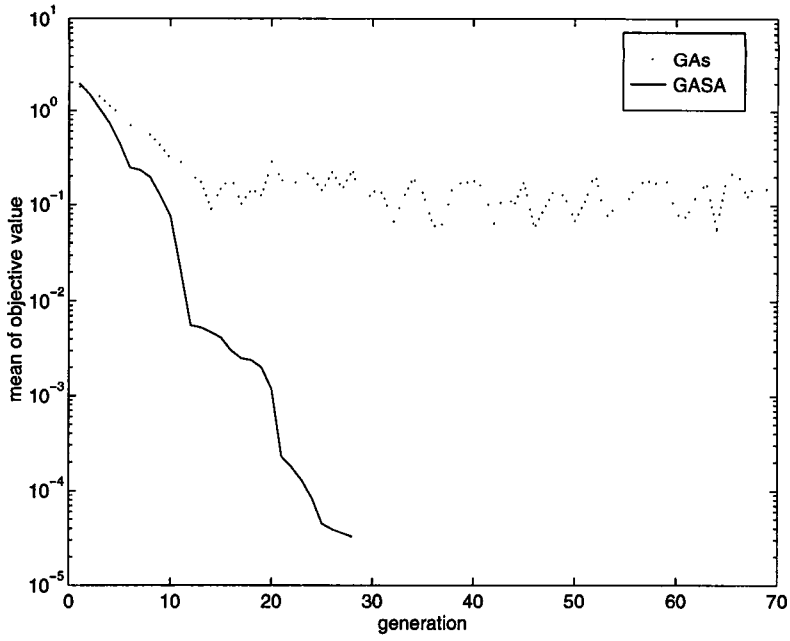


**Figure 9**: *Mean objective value in generations*

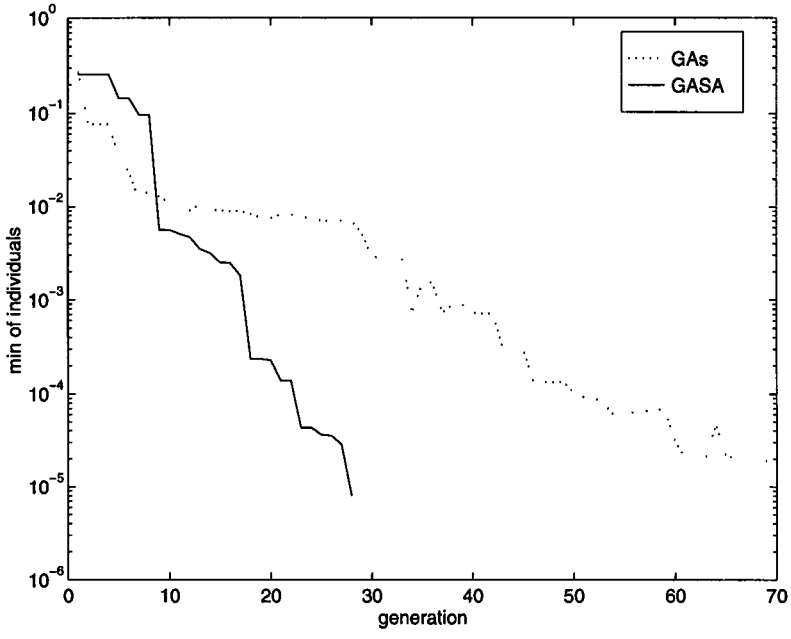


**Figure 10**: *Best objective value in generations*

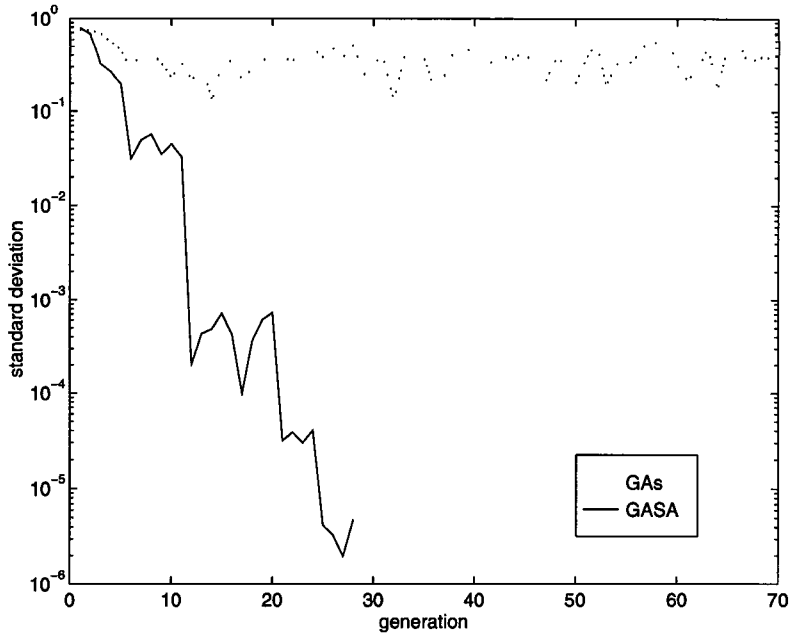


**Figure 11**: *Standard deviation of population vs. generation*

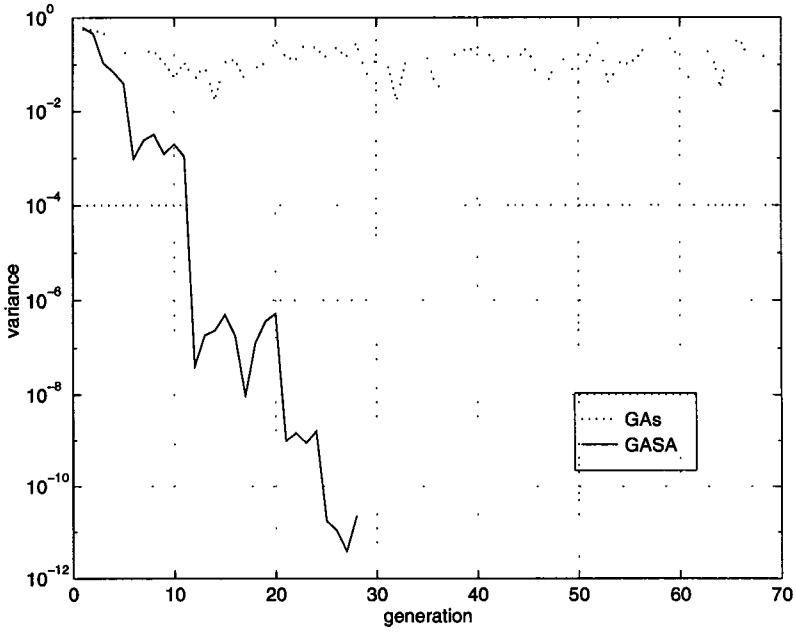


**Figure 12**: *Variance of population vs. generation*

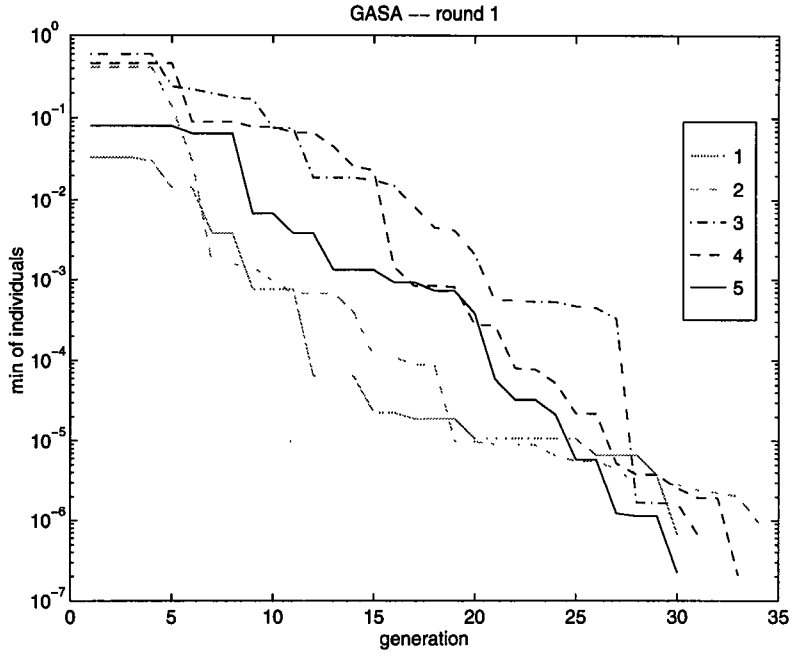


**Figure 13**: *The minimum of indviduals vs. generations in 1∼5 sets of GASA in round 1*

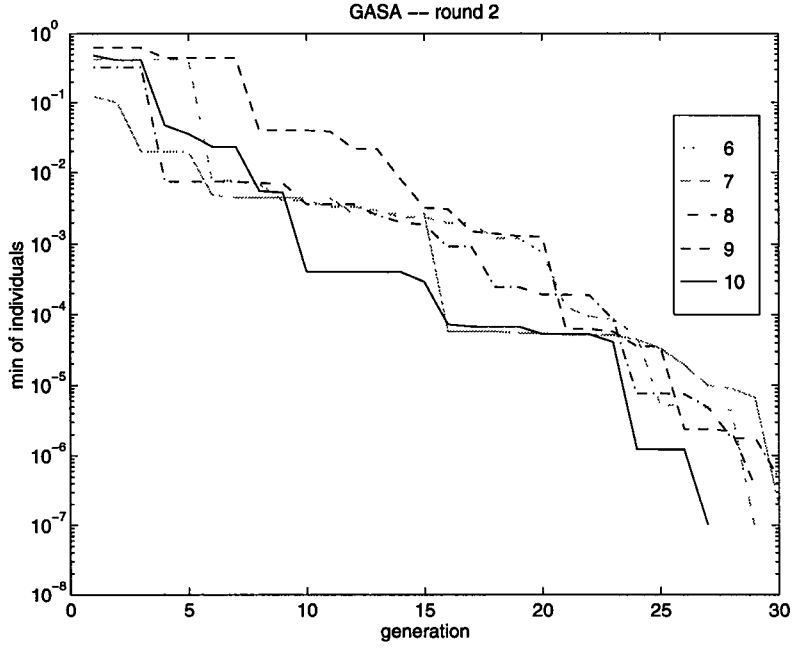


**Figure 14**: *The minimum of indviduals vs. generations in 6∼10 sets of GASA in round 2*

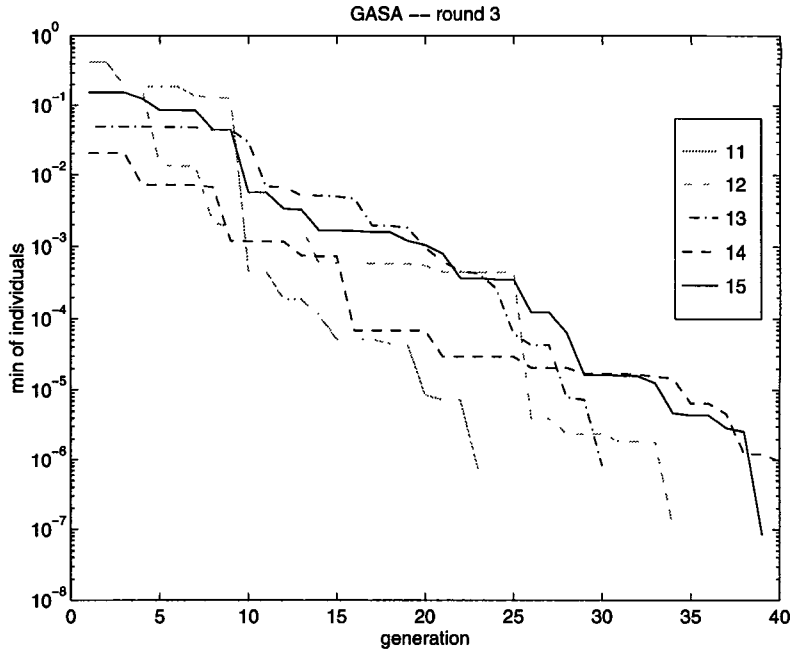


**Figure 15**: *The minimum of indviduals vs. generations in 11~15 sets of GASA in round 3*

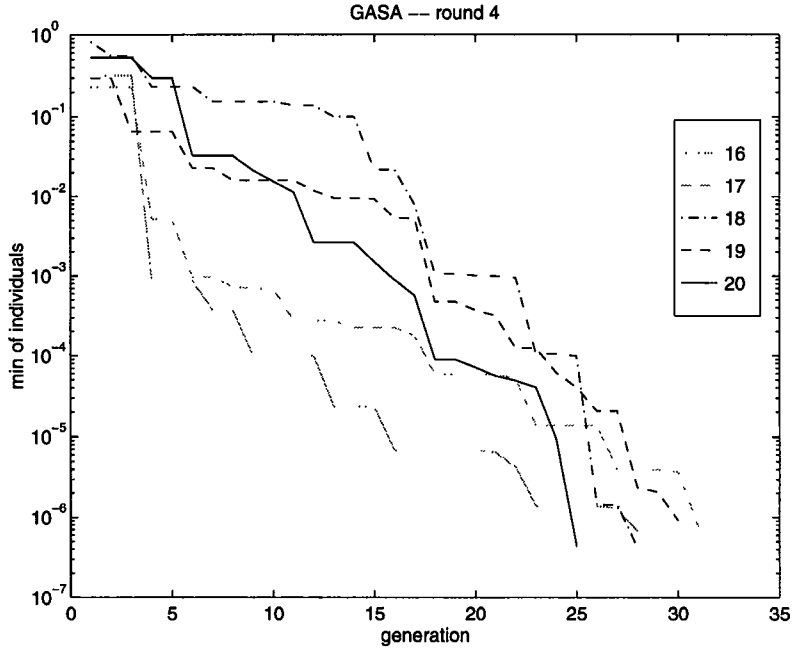


**Figure 16**: *The minimum of indviduals vs. generations in 16~20 sets of GASA in round 4*

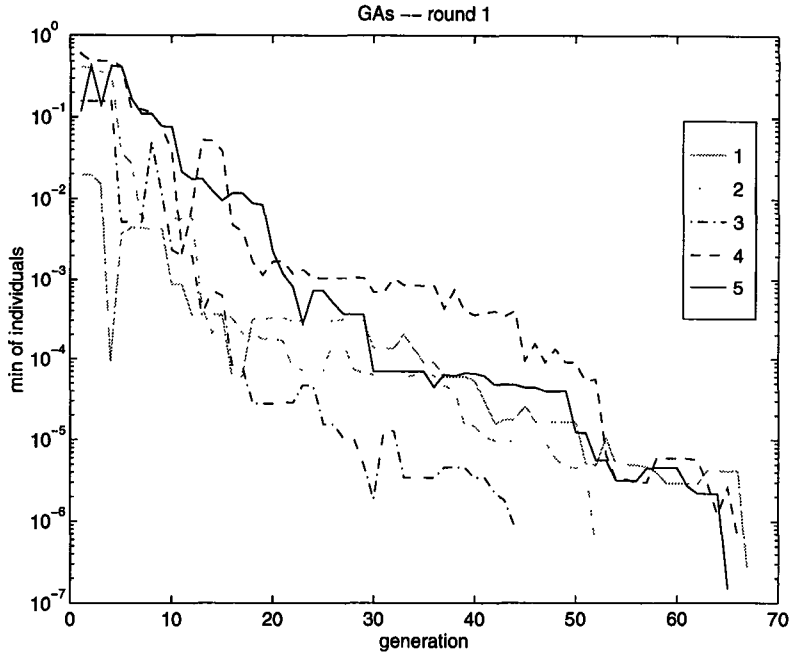


**Figure 17**: *The minimum of indviduals vs. generations in 1∼5 sets of GAs of round 1*

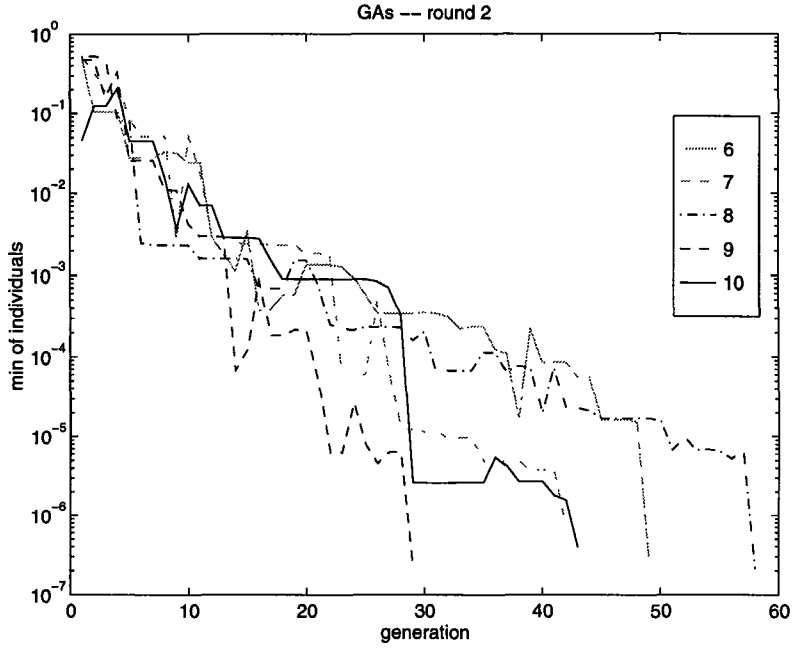


**Figure 18**: *The minimum of indviduals vs. generations in 6∼10 sets of GAs in round 2*

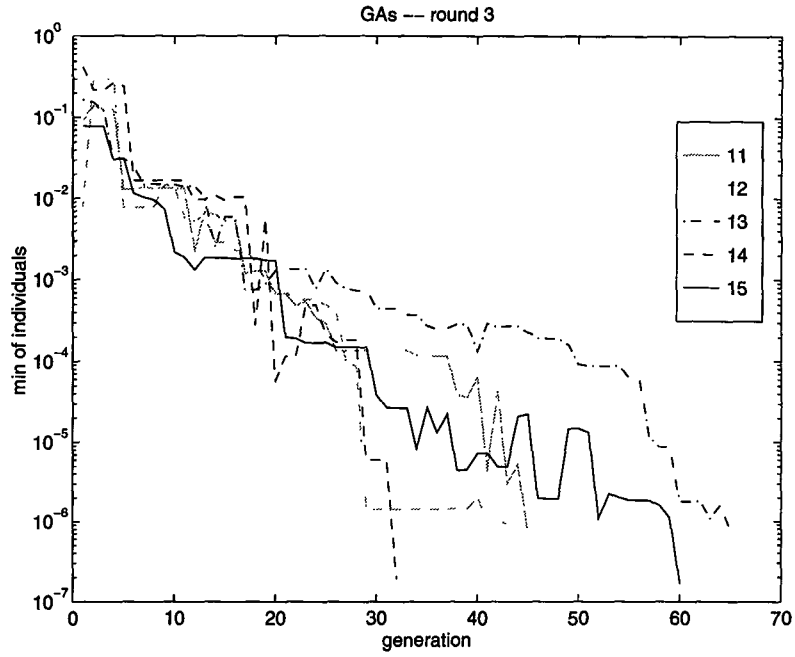


**Figure 19**: *The minimum of indviduals vs. generations in 11~15 sets of GAs of round 3*

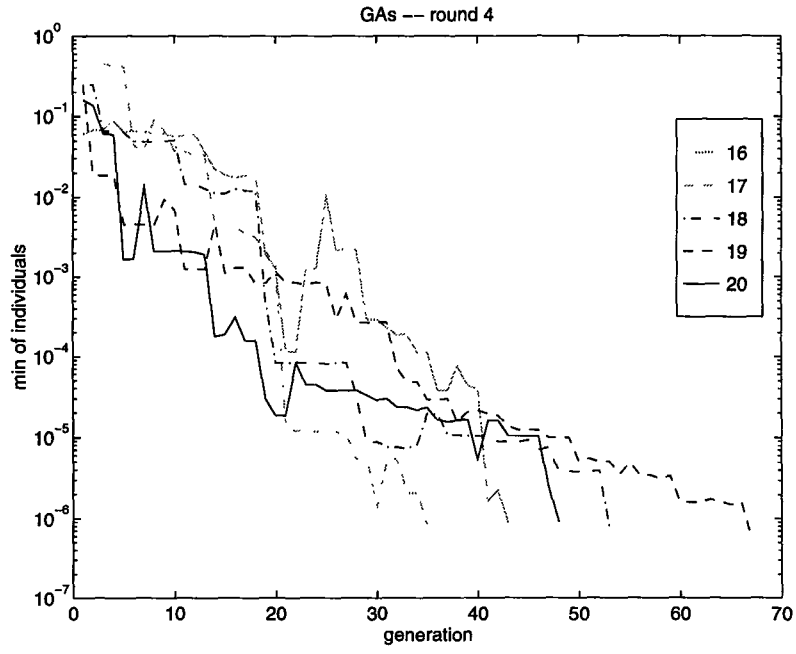


**Figure 20**: *The minimum of indviduals vs. generations in 16~20 sets of GAs in round 4*

# References


[1] S. Arunkumar and T. Chockalingam. Genetic search algorithms and their randomized operators. *Computers & Mathematics with Applications*, 25(5):91–100, March 1993.

[2] Ihor O. Bohachevsky. Generalized simulated annealing for function optimization. *TECHNOMETRICS*, 28(3):209–217, 1986.

[3] Thomas Dandekar and Patrick Argos. Folding the main chains of small proteins with the genetic algorithm. *Journal of Molecular Biology*, 236:844–861, 1994.

[4] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[5] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-(6):721–741, 1984.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Maching Learning*. Addision-Wesley, Reading, MA, 1989.

[7] D. E. Goldberg and R. Lingle. Alleles, loci, and the traveling salesman problem. In Grefenstette [8], pages 154–159. Canegie-Mellon Univ.

[8] J. J. Grefenstette. *Proceedings of an International Conference on Genetic Algorithms and Their Applications*. Lawerence Erlbaum Associates, Hilldale, NJ, 1985a.

[9] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press., Ann Arbor, 1975.

[10] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4958):671–680, 1983.

[11] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4(4):461–476, August 1990.

[12] F. T. Lin, C. Y. Kao, and C. C. Hsu. Applied genetic approach to simulated annealing in solving some np-hard probelms. *IEEE Transactions on Systems, Man, & Cybernetics*, 23(6):1752–1767, 1993.

[13] N. Metropolis, A. W. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. of Chem. Phys.*, 21(6):1087–1092, 1953.

[14] Zbigniew Michalewich. *Genetic Algorithms + Data Structures = Evoultionary Programs*. Springer-Verlag, Berlin Heidelberg, Germany, 1992.

[15] John A. Miller, Walter D. Potter, Ravi V. Gandham, and Chito N. Lapena. An evaluation of local improvement operators for genetic algorithms. *IEEE Transactions on Systems, Man, & Cybernetics*, 23(5):1340–1351, September/October 1993.

[16] D. Park, A. Kandel, and G. Langholz. Genetic based new fuzzy reasoning models with application to fuzzy control. *IEEE Transactions on Systems, Man, & Cybernetics*, 24(1):39–47, 1994.

[17] J. Craig Potts, Terri D. Giddens, and Surya B. Yadav. The development and evaluation of an improved genetic algorithm based on migration and artificial selection. *IEEE Transactions on Systems, Man, & Cybernetics*, 1994.

[18] D. J. Sirag and P. T. Weisser. Torward a unified thermodynamic operator. *Proc. of 2nd Intl. Conf. of Genetic Algorithms and Their Applications*, pages 116–122, 1987.

[19] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231(1):75–81, May 1993.

[20] Alen Varšek, Tanja Urbančič, and Bodgan Filipič. Genetic algorithms in controller design and tuning. *SMC*, 23(5):1330–1339, September/October 1993.

[21] R. Wehrens, C. Lucasius, L. Buydens, and G. Kateman. Sequential assignment of 2D-NMR spectra of proteins using genetic algorithms. *Journal of Chemical Information and Computer Sciences*, 33(2):245–251, Mar-Apr 1993.