Evolutionary Programming Made Faster*†

Xin Yao, Yong Liu, and Guangming Lin
Computational Intelligence Group, School of Computer Science
University College, The University of New South Wales
Australian Defence Force Academy, Canberra, ACT, Australia 2600
Email: {xin,liuy,glin}@cs.adfa.edu.au, URL: http://www.cs.adfa.edu.au/~xin

Abstract

Evolutionary programming (EP) has been applied with success to many numerical and combinatorial optimization problems in recent years. However, EP has rather slow convergence rates on some function optimization problems. In this paper, a "fast EP" (FEP) is proposed which uses a Cauchy instead of Gaussian mutation as the primary search operator. The relationship between FEP and classical EP (CEP) is similar to that between the fast simulated annealing and the classical version. Both analytical and empirical studies have been carried out to evaluate the performance of FEP and CEP for different function optimization problems. This paper shows that FEP is very good at search in a large neighborhood while CEP is better at search in a small local neighborhood. For a suite of 23 benchmark problems, FEP performs much better than CEP for multimodal functions with many local minima while being comparable to CEP in performance for unimodal and multimodal functions with only a few local minima. This paper also shows the relationship between the search step size and the probability of finding a global optimum, and thus explains why FEP performs better than CEP on some functions but not on others. In addition, the importance of the neighborhood size and its relationship to the probability of finding a near-optimum is investigated. Based on these analyses, an improved FEP (IFEP) is proposed and tested empirically. This technique mixes different search operators (mutations). The experimental results show that IFEP performs better than or as well as the better of FEP and CEP for most benchmark problems tested.

1 Introduction

Although evolutionary programming (EP) was first proposed as an approach to artificial intelligence [1], it has been recently applied with success to many numerical and combinatorial optimization problems [2, 3, 4]. Optimization by EP can be summarized into two major steps:

1. Mutate the solutions in the current population, and
2. Select the next generation from the mutated and the current solutions.

These two steps can be regarded as a population-based version of the classical generate-and-test method [5], where mutation is used to **generate** new solutions (offspring) and selection is used to **test** which of the newly generated solutions should survive to the next generation. Formulating EP as a special case of the generate-and-test method establishes a bridge between EP and other search algorithms, such as evolution strategies, genetic algorithms, simulated annealing (SA), tabu search (TS), and others, and thus facilitates cross-fertilization among different research areas.

*This work is partially supported by the Australian Research Council through its small grant scheme and by a special research grant from the University College, UNSW, ADFA.

†Published as: X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, **3**(2):82-102, July 1999.

One disadvantage of EP in solving some of the multimodal optimization problems is its slow convergence to a good near-optimum (e.g., f_8 to f_{13} studied in this paper). The generate-and-test formulation of EP indicates that mutation is a key search operator which generates new solutions from the current ones. A new mutation operator based on Cauchy random numbers is proposed and tested on a suite of 23 functions in this paper. The new EP with Cauchy mutation significantly outperforms the classical EP (CEP), which uses Gaussian mutation, on a number of multimodal functions with many local minima while being comparable to CEP for unimodal and multimodal functions with only a few local minima. The new EP is denoted as "fast EP" (FEP) in this paper.

Extensive empirical studies of both FEP and CEP have been carried out in order to evaluate the relative strength and weakness of FEP and CEP for different problems. The results show that Cauchy mutation is an efficient search operator for a large class of multimodal function optimization problems. FEP's performance can be expected to improve further since all the parameters used in the FEP were set equivalently to those used in CEP.

In order to explain why Cauchy mutation performs better than Gaussian mutation for most benchmark problems used here, theoretical analysis has been carried out to show the importance of the neighborhood size and search step size in EP. It is shown that Cauchy mutation performs better because of its higher probability of making longer jumps. Although the idea behind FEP appears to be simple and straightforward (the larger the search step size, the faster the algorithm gets to the global optimum), no theoretical result has been provided to answer the question why this is the case and how fast it might be. In addition, a large step size may not be beneficial at all if the current search point is already very close to the global optimum. This paper shows for the first time the relationship between the distance to the global optimum and the search step size, and the relationship between the search step size and the probability of finding a near (global) optimum. Based on such analyses, an improved FEP has been proposed and tested empirically.

The rest of this paper is organized as follows. Section 2 describes the global minimization problem considered in this paper and the CEP used to solve it. The CEP algorithm given follows suggestions from Fogel [3, 6] and Bäck and Schwefel [7]. Section 3 describes the FEP and its implementation. Section 4 gives the 23 functions used in our studies. Section 5 presents the experimental results and discussions on FEP and CEP. Section 6 investigates FEP with different scale parameters for its Cauchy mutation. Section 7 analyzes FEP and CEP and explains the performance difference between FEP and CEP in depth. Based on such analyses, an improved FEP (IFEP) is proposed and tested in Section 8. Finally, Section 9 concludes with some remarks and future research directions.

2 Function Optimization By Classical Evolutionary Programming

A global minimization problem can be formalized as a pair (S, f), where $S \subseteq R^n$ is a bounded set on R^n and $f : S \mapsto R$ is an n-dimensional real-valued function. The problem is to find a point $\mathbf{x}_{min} \in S$ such that $f(\mathbf{x}_{min})$ is a global minimum on S. More specifically, it is required to find an $\mathbf{x}_{min} \in S$ such that

$$\forall \mathbf{x} \in S : f(\mathbf{x}_{min}) \leq f(\mathbf{x}),$$

where f does not need to be continuous but it must be bounded. This paper only considers unconstrained function optimization.

Fogel [3, 8] and Bäck and Schwefel [7] have indicated that CEP with self-adaptive mutation usually performs better than CEP without self-adaptive mutation for the fuctions they tested. Hence the CEP with self-adaptive mutation will be investigated in this paper. According to the description by Bäck and Schwefel [7], the CEP is implemented as follows in this study:[1]

[1]A recent study by Gehlhaar and Fogel [9] showed that swapping the order of Eq.1 and Eq.2 may improve CEP's performance.

1. Generate the initial population of μ individuals, and set $k = 1$. Each individual is taken as a pair of real-valued vectors, $(\mathbf{x}_i, \eta_i)$, $\forall i \in \{1, \cdots, \mu\}$, where $\mathbf{x}_i$'s are objective variables and η_i's are standard deviations for Gaussian mutations (also known as strategy parameters in self-adaptive evolutionary algorithms).

2. Evaluate the fitness score for each individual $(\mathbf{x}_i, \eta_i)$, $\forall i \in \{1, \cdots, \mu\}$, of the population based on the objective function, $f(\mathbf{x}_i)$.

3. Each parent $(\mathbf{x}_i, \eta_i)$, $i = 1, \cdots, \mu$, creates a single offspring $(\mathbf{x}_i', \eta_i')$ by: for $j = 1, \cdots, n$,

$$\begin{aligned} x_i'(j) &= x_i(j) + \eta_i(j) N_j(0,1), && (1) \\ \eta_i'(j) &= \eta_i(j) \exp(\tau' N(0,1) + \tau N_j(0,1)), && (2) \end{aligned}$$

   where $x_i(j)$, $x_i'(j)$, $\eta_i(j)$ and $\eta_i'(j)$ denote the j-th component of the vectors $\mathbf{x}_i$, $\mathbf{x}_i'$, η_i and η_i', respectively. $N(0,1)$ denotes a normally distributed one-dimensional random number with mean 0 and standard deviation 1. $N_j(0,1)$ indicates that the random number is generated anew for each value of j. The factors τ and τ' are commonly set to $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$ [7, 6].

4. Calculate the fitness of each offspring $(\mathbf{x}_i', \eta_i')$, $\forall i \in \{1, \cdots, \mu\}$.

5. Conduct pairwise comparison over the union of parents $(\mathbf{x}_i, \eta_i)$ and offspring $(\mathbf{x}_i', \eta_i')$, $\forall i \in \{1, \cdots, \mu\}$. For each individual, q opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a "win."

6. Select the μ individuals out of $(\mathbf{x}_i, \eta_i)$ and $(\mathbf{x}_i', \eta_i')$, $\forall i \in \{1, \cdots, \mu\}$, that have the most wins to be parents of the next generation.

7. Stop if the halting criterion is satisfied; otherwise, $k = k + 1$ and go to Step 3.

3 Fast Evolutionary Programming

The one-dimensional Cauchy density function centered at the origin is defined by:

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \qquad -\infty < x < \infty, \tag{3}$$

where $t > 0$ is a scale parameter [10](pp.51). The corresponding distribution function is

$$F_t(x) = \frac{1}{2} + \frac{1}{\pi} arctan\left(\frac{x}{t}\right).$$

The shape of $f_t(x)$ resembles that of the Gaussian density function but approaches the axis so slowly that an expectation does not exist. As a result, the variance of the Cauchy distribution is infinite. Figure 1 shows the difference between Cauchy and Gaussian functions by plotting them in the same scale.

The FEP studied in this paper is exactly the same as the CEP described in Section 2 except for Eq.(1) which is replaced by the following [11]:

$$x_i'(j) = x_i(j) + \eta_i(j)\delta_j, \tag{4}$$

where δ_j is a Cauchy random variable with the scale parameter $t = 1$, and is generated anew for each value of j. It is worth indicating that we leave Eq.(2) unchanged in FEP in order to keep

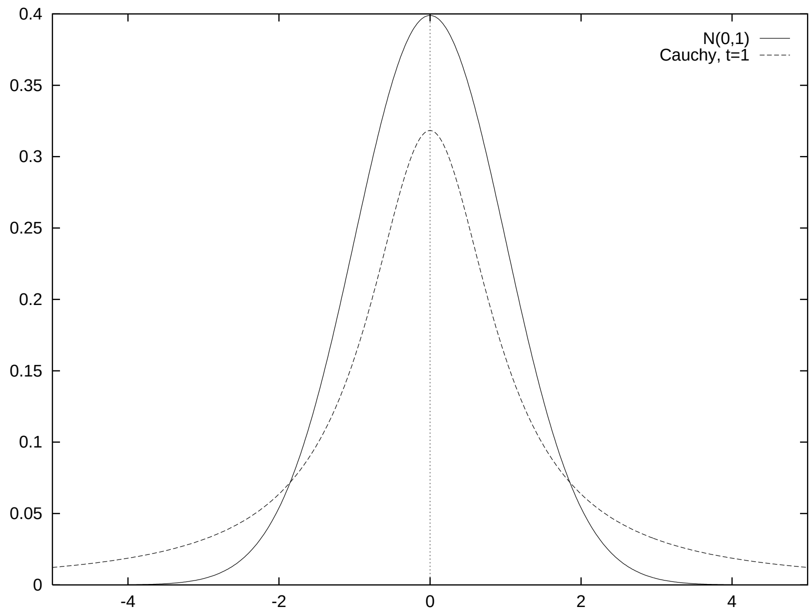


Figure 1: Comparison between Cauchy and Gaussian density functions.

our modification of CEP to a minimum. It is also easy to investigate the impact of the Cauchy mutation on EP when other parameters are kept the same.

It is clear from Figure 1 that Cauchy mutation is more likely to generate an offspring further away from its parent than Gaussian mutation due to its long flat tails. It is expected to have a higher probability of escaping from a local optimum or moving away from a plateau, especially when the "basin of attraction" of the local optimum or the plateau is large relative to the mean step size. On the other hand, the smaller hill around the center in Figure 1 indicates that Cauchy mutation spends less time in exploiting the local neighborhood and thus has a weaker fine-tuning ability than Gaussian mutation in small to mid-range regions. Our empirical results support the above intuition.

4 Benchmark Functions

Twenty-three benchmark functions [2, 7, 12, 13] were used in our experimental studies. This number is larger than that offered in many other empirical study papers. However, this is necessary since the aim here is not to show FEP is better or worse than CEP, but to find out when FEP is better (or worse) than CEP and why. Wolpert and Macready [14, 15] have shown that under certain assumptions no single search algorithm is best on average for all problems. If the number of test problems is small, it would be very difficult to make a generalized conclusion. Using too small a test set also has the potential risk that the algorithm is biased (optimized) towards the chosen problems, while such bias might not be useful for other problems of interest.

The 23 benchmark functions are given in Table 1. More detailed description of each function is given in the appendix. Functions f_1 to f_{13} are high-dimensional problems. Functions f_1 to f_5 are unimodal. Function f_6 is the step function, which has one minimum and is discontinuous. Function f_7 is a noisy quartic function, where $random[0, 1)$ is a uniformly distributed random variable in $[0, 1)$. Functions f_8 to f_{13} are multimodal functions where the number of local minima increases exponentially with the problem dimension [12, 13]. They appear to be the most difficult class of problems for many optimization algorithms (including EP). Functions f_{14} to f_{23} are low-dimensional functions which have only a few local minima [12]. For unimodal functions, the convergence rates of FEP and CEP are more interesting than the final results of optimization as there are other methods which are specifically designed to optimize unimodal functions. For multimodal functions, the final results are much more important since they reflect an algorithm's ability of escaping from poor local optima and locating a good near-global optimum.

5 Experimental Studies

5.1 Experimental Setup

In all experiments, the same self-adaptive method (i.e., Eq.(2)), the same population size $\mu = 100$, the same tournament size $q = 10$ for selection, the same initial $\eta = 3.0$, and the same initial population were used for both CEP and FEP. These parameters follow the suggestions from Bäck and Schwefel [7] and Fogel [2]. The initial population was generated uniformly at random in the range as specified in Table 1.

5.2 Unimodal Functions

The first set of experiments was aimed to compare the convergence rate of CEP and FEP for functions f_1 to f_7. The average results of 50 independent runs are summarized in Table 2. Figures 2 and 3 show the progress of the mean best solutions and the mean of average values of population found by CEP and FEP over 50 runs for f_1 to f_7. It is apparent that FEP performs better than CEP in terms of convergence rate although CEP's final results were better than FEP's for functions f_1 and f_2. Function f_1 is the simple sphere model studied by many researchers. CEP's and FEP's behaviors on f_1 are quite illuminating. In the beginning, FEP dispays a faster convergence rate than CEP due to its better global search ability. It quickly approaches the neighborhood of the global optimum and reaches approximately 0.001 in around 1200 generations, while CEP can only reach approximately 0.1. After that, FEP's convergence rate reduces substantially, while CEP maintains a nearly constant convergence rate throughout the evolution. Finally, CEP overtakes FEP at around 1450th generation. As indicated in Section 3 and in Figure 1, FEP is weaker than CEP in fine-tuning. Such weakness slows down its convergence considerably in the neighborhood of the global optimum. CEP is capable of maintaining its nearly constant convergence rate because its search is much more localized than FEP. The different behavior of CEP and FEP on f_1 suggests that CEP is better at fine-grained search while FEP is better at coarse-grained search.

The largest difference in performance between CEP and FEP occurs with function f_6, the step function, which is characterized by plateaus and discontinuity. CEP performs poorly on the step function because it mainly searches in a relatively small local neighborhood. All the points within the local neighborhood will have the same fitness value except for a few boundaries between plateaus. Hence it is very difficult for CEP to move from one plateau to a lower one. On the other hand, FEP has a much higher probability of generating long jumps than CEP. Such long jumps enable FEP to move from one plateau to a lower one with relative ease. The rapid convergence of FEP shown in Figure 3 supports our explanations.

5.3 Multimodal Functions

5.3.1 Multimodal Functions With Many Local Minima

Multimodal functions having many local minima are often regarded as being difficult to optimize. f_8–f_{13} are such functions where the number of local minima increases exponentially as the dimension of the function increases. Figure 4 shows the 2-dimensional version of f_8.

The dimensions of f_8–f_{13} were all set to 30 in our experiments. Table 3 summarizes the final results of CEP and FEP. It is obvious that FEP performs significantly better than CEP consistently for these functions. CEP appeared to become trapped in a poor local optimum and unable to escape from it due to its smaller probability of making long jumps. According to the figures we plotted in

Table 1: The 23 benchmark functions used in our experimental study, where n is the dimension of the function, f_{min} is the minimum value of the function, and $S \subseteq R^n$. A detailed description of all functions is given in the Appendix.

Test function	n	S	f_{min}
$f_1(x) = \sum_{i=1}^n x_i^2$	30	$[-100, 100]^n$	0
$f_2(x) = \sum_{i=1}^n \lvert x_i\rvert + \prod_{i=1}^n \lvert x_i\rvert$	30	$[-10, 10]^n$	0
$f_3(x) = \sum_{i=1}^n (\sum_{j=1}^i x_j)^2$	30	$[-100, 100]^n$	0
$f_4(x) = \max_i\{\lvert x_i\rvert, 1 \le i \le n\}$	30	$[-100, 100]^n$	0
$f_5(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]^n$	0
$f_6(x) = \sum_{i=1}^n (\lfloor x_i + 0.5\rfloor)^2$	30	$[-100, 100]^n$	0
$f_7(x) = \sum_{i=1}^n ix_i^4 + random[0, 1)$	30	$[-1.28, 1.28]^n$	0
$f_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{\lvert x_i\rvert})$	30	$[-500, 500]^n$	-12569.5
$f_9(x) = \sum_{i=1}^n [x_i^2 - 10\cos(2\pi x_i) + 10)]$	30	$[-5.12, 5.12]^n$	0
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^n \cos 2\pi x_i\right)$ $+20 + e$	30	$[-32, 32]^n$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^n$	0
$f_{12}(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]\right.$ $\left.+(y_n - 1)^2\right\} + \sum_{i=1}^n u(x_i, 10, 100, 4),$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	30	$[-50, 50]^n$	0
$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + sin^2(3\pi x_{i+1})]\right.$ $\left.+(x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	30	$[-50, 50]^n$	0
$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6}\right]^{-1}$	2	$[-65.536, 65.536]^n$	1
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]^n$	0.0003075
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^n$	-1.0316285
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, 10] \times [0, 15]$	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2$ $+6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1$ $+12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]^n$	3
$f_{19}(x) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^4 a_{ij}(x_j - p_{ij})^2\right]$	4	$[0, 1]^n$	-3.86
$f_{20}(x) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2\right]$	6	$[0, 1]^n$	-3.32
$f_{21}(x) = -\sum_{i=1}^5[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10
$f_{22}(x) = -\sum_{i=1}^7[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10
$f_{23}(x) = -\sum_{i=1}^{10}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10

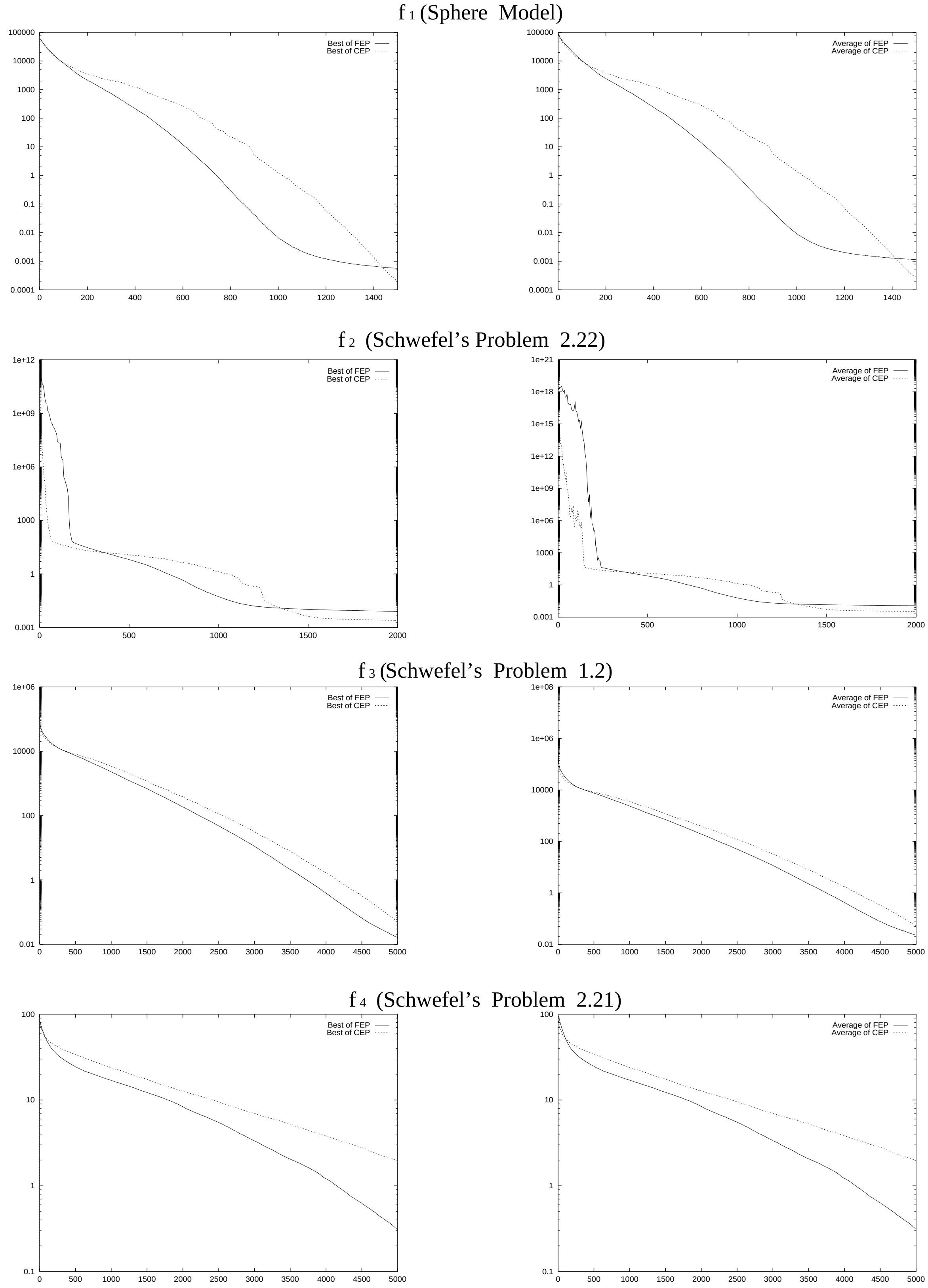
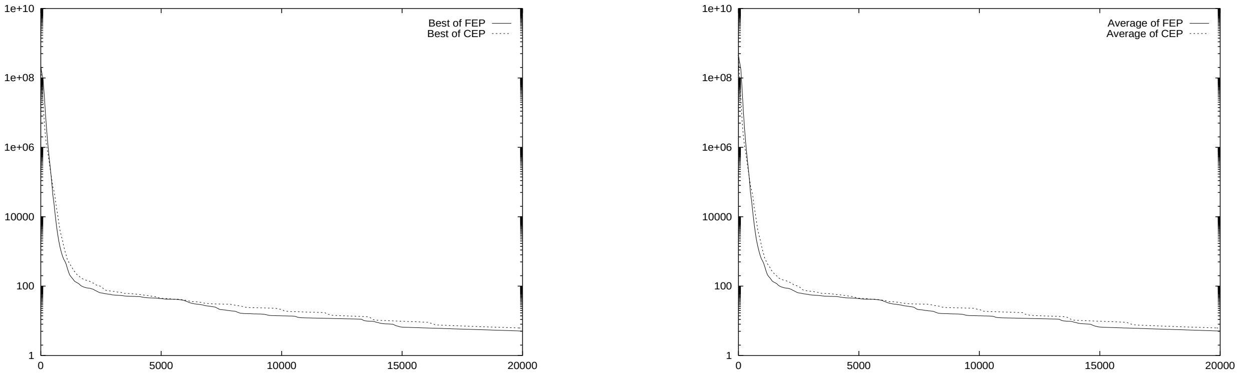
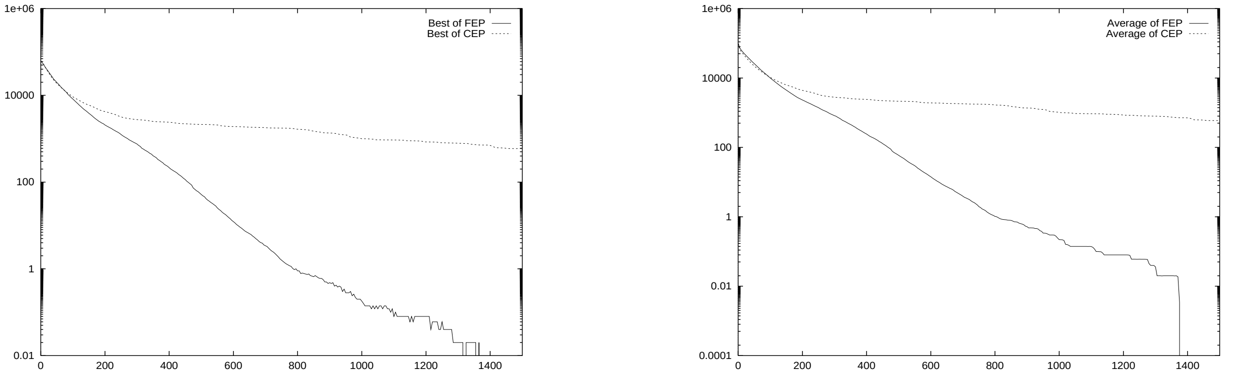


Figure 2: Comparison between CEP and FEP on f_1–f_4. The vertical axis is the function value and the horizontal axis is the number of generations. The solid lines indicate the results of FEP. The dotted lines indicate the results of CEP. The left figures show the best results. The right figures show the average results. Both were averaged over 50 runs.

f 5 (Generalized Rosenbrock's Function)



f 6 (Step Function)



f 7 (Quartic Function with Noise)

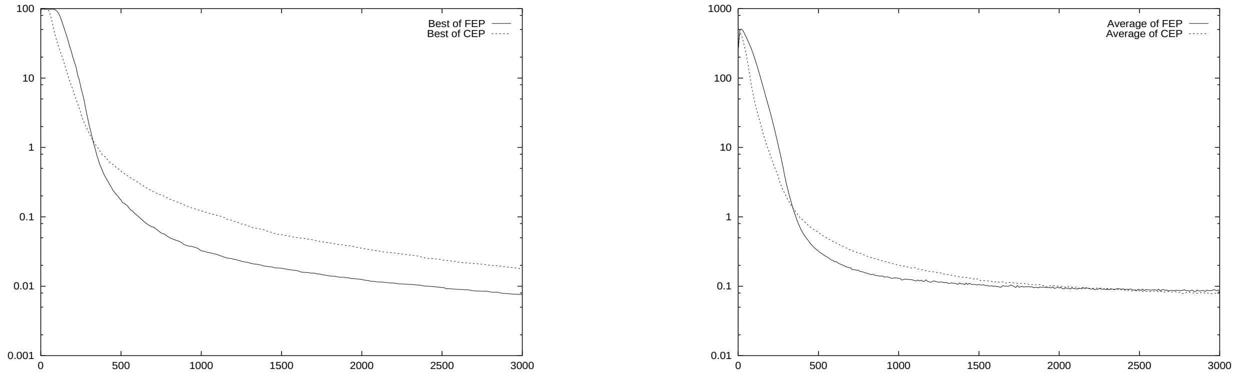


Figure 3: Comparison between CEP and FEP on f_5–f_7.The vertical axis is the function value and the horizontal axis is the number of generations. The solid lines indicate the results of FEP. The dotted lines indicate the results of CEP. The left figures show the best results. The right figures show the average results. Both were averaged over 50 runs.

Table 2: Comparison between CEP and FEP on f_1–f_7. All results have been averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation, and "Std Dev" stands for the standard deviation.

Function	Number of	FEP		CEP		FEP−CEP
	Generations	Mean Best	Std Dev	Mean Best	Std Dev	t-test
f_1	1500	5.7×10^{-4}	1.3×10^{-4}	2.2×10^{-4}	5.9×10^{-4}	$4.06^{\dagger}$
f_2	2000	8.1×10^{-3}	7.7×10^{-4}	2.6×10^{-3}	1.7×10^{-4}	$49.83^{\dagger}$
f_3	5000	1.6×10^{-2}	1.4×10^{-2}	5.0×10^{-2}	6.6×10^{-2}	$-3.79^{\dagger}$
f_4	5000	0.3	0.5	2.0	1.2	$-8.25^{\dagger}$
f_5	20000	5.06	5.87	6.17	13.61	-0.52
f_6	1500	0	0	577.76	1125.76	$-3.67^{\dagger}$
f_7	3000	7.6×10^{-3}	2.6×10^{-3}	1.8×10^{-2}	6.4×10^{-3}	$-10.72^{\dagger}$

†The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

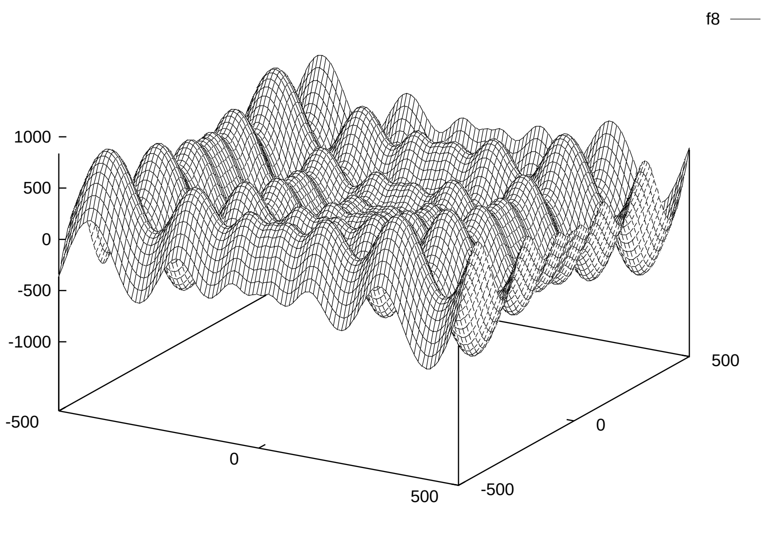


Figure 4: The 2-dimensional version of f_8.

order to observe the evolutionary process, CEP fell into a poor local optimum quite early in a run while FEP was able to improve its solution steadily for a long time. FEP appeared to converge at least at a linear rate with respect to the number of generations. An exponential convergence rate was observed for some problems.

Table 3: Comparison between CEP and FEP on f_8–f_{13}. The results are averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation, and "Std Dev" stands for the standard deviation.

Function	Number of	FEP		CEP		FEP−CEP
	Generations	Mean Best	Std Dev	Mean Best	Std Dev	t-test
f_8	9000	−12554.5	52.6	−7917.1	634.5	−51.39†
f_9	5000	4.6×10^{-2}	1.2×10^{-2}	89.0	23.1	−27.25†
f_{10}	1500	1.8×10^{-2}	2.1×10^{-3}	9.2	2.8	−23.33†
f_{11}	2000	1.6×10^{-2}	2.2×10^{-2}	8.6×10^{-2}	0.12	−4.28†
f_{12}	1500	9.2×10^{-6}	3.6×10^{-6}	1.76	2.4	−5.29†
f_{13}	1500	1.6×10^{-4}	7.3×10^{-5}	1.4	3.7	−2.76†

†The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

5.3.2 Multimodal Functions With Only a Few Local Minima

In order to evaluate FEP more fully, additional multimodal benchmark functions were also included in our experiments, i.e., f_{14}–f_{23}, where the number of local minima for each function and the dimension of the function are small. Table 4 summarizes the results averaged over 50 runs.

Table 4: Comparison between CEP and FEP on f_{14}–f_{23}. The results are averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation, and "Std Dev" stands for the standard deviation.

Function	Number of	FEP		CEP		FEP−CEP
	Generations	Mean Best	Std Dev	Mean Best	Std Dev	t-test
f_{14}	100	1.22	0.56	1.66	1.19	−2.21†
f_{15}	4000	5.0×10^{-4}	3.2×10^{-4}	4.7×10^{-4}	3.0×10^{-4}	0.49
f_{16}	100	−1.03	4.9×10^{-7}	−1.03	4.9×10^{-7}	0.0
f_{17}	100	0.398	1.5×10^{-7}	0.398	1.5×10^{-7}	0.0
f_{18}	100	3.02	0.11	3.0	0	1.0
f_{19}	100	−3.86	1.4×10^{-5}	−3.86	1.4×10^{-2}	−1.0
f_{20}	200	−3.27	5.9×10^{-2}	−3.28	5.8×10^{-2}	0.45
f_{21}	100	−5.52	1.59	−6.86	2.67	3.56†
f_{22}	100	−5.52	2.12	−8.27	2.95	5.44†
f_{23}	100	−6.57	3.14	−9.10	2.92	4.24†

†The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

Interestingly, quite different results have been observed for functions f_{14} to f_{23}. For 6 (i.e., f_{15}–f_{20}) out of 10 functions, no statistically significant difference was found between FEP and CEP. In fact, FEP performed exactly the same as CEP for f_{16} and f_{17}. For the four functions where there was statistically significant difference between FEP and CEP, FEP performed better for f_{14}, but was outperformed by CEP for f_{21}–f_{23}. The consistent superiority of FEP over CEP for functions f_8–f_{13} was not observed here.

The major difference between functions f_8–f_{13} and f_{14}–f_{23} is that functions f_{14}–f_{23} appear to be simpler than f_8–f_{13} due to their low dimensionalities and a smaller number of local minima. In order to find out whether or not the dimensionality of functions plays a significant role in deciding FEP's and CEP's behavior, another set of experiments on the low-dimensional ($n = 5$) version of f_8–f_{13} was carried out. The results averaged over 50 runs are given in Table 5.

Table 5: Comparison between CEP and FEP on f_8 to f_{13} with $n = 5$. The results are averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation, and "Std Dev" stands for the standard deviation.

Function	Number of Generations	FEP Mean Best	FEP Std Dev	CEP Mean Best	CEP Std Dev	FEP−CEP t-test
f_8	500	-2061.74	58.79	-1762.45	176.21	$-11.17^\dagger$
f_9	400	0.14	0.40	4.08	3.08	$-8.89^\dagger$
f_{10}	400	8.6×10^{-4}	1.8×10^{-4}	8.1×10^{-2}	0.34	-1.67
f_{11}	1500	5.3×10^{-2}	4.2×10^{-2}	0.14	0.12	$-4.64^\dagger$
f_{12}	200	1.5×10^{-7}	1.2×10^{-7}	2.5×10^{-2}	0.12	-1.43
f_{13}	200	3.5×10^{-7}	1.8×10^{-7}	3.8×10^{-3}	1.4×10^{-2}	-1.89

†The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

Very similar results to the previous ones on functions f_8–f_{13} were obtained despite the large difference in the dimensionality of functions. FEP still outperforms CEP significantly even when the dimensionality of functions f_8–f_{13} is low ($n = 5$). It is clear that dimensionality is not the key factor which determines the behavior of FEP and CEP. It is the shape of the function and/or the number of local minima that have a major impact on FEP's and CEP's performance.

6 Fast Evolutionary Programming With Different Parameters

FEP investigated in the previous section used $t = 1$ in its Cauchy mutation. This value was used for its simplicity. In order to examine the impact of different t values on the performance of FEP in detail, a set of experiments have been carried out on FEP using different t values for the Cauchy mutation. Seven benchmark functions from the three different groups in Table 1 were used in these experiments. The setup of these experiments is exactly the same as before. Table 6 shows the average results over 50 independent runs of FEP for different parameters.

These results show that $t = 1$ was not the optimal value for the seven benchmark problems. The optimal t is problem-dependent. As analyzed later in Section 7.1, the optimal t depends on the distance between the current search point and the global optimum. Since the global optimum is usually unknown for real-world problems, it is extremely difficult to find the optimal t for a given problem. A good approach to deal with this issue is to use self-adaptation so that t can gradually evolve towards its near optimum although its initial value might not be optimal.

Another approach to be explored is to mix Cauchy mutation with different t values in a population so that the whole population can search both globally and locally. The percentage of each type of Cauchy mutation will be self-adaptive, rather than fixed. Hence the population may emphasize either global or local search depending on different stages in the evolutionary process.

Table 6: The mean best solutions found by FEP using different scale parameter t in the Cauchy mutation for functions $f_1(1500)$, $f_2(2000)$, $f_{10}(1500)$, $f_{11}(2000)$, $f_{21}(100)$, $f_{22}(100)$ and $f_{23}(100)$. The values in "()" indicate the number of generations used in FEP. All results have been averaged over 50 runs.

Function	$t = 0.0156$	$t = 0.0313$	$t = 0.0625$	$t = 0.1250$	$t = 0.2500$
f_1	1.0435	0.0599	0.0038	1.5×10^{-4}	6.5×10^{-5}
f_2	3.8×10^{-4}	3.1×10^{-4}	5.9×10^{-4}	0.0011	0.0021
f_{10}	1.5627	0.2858	0.0061	0.0030	0.0050
f_{11}	1.0121	0.2237	0.1093	0.0740	0.0368
f_{21}	−6.9236	−7.7261	−8.0487	−8.6473	−8.0932
f_{22}	−7.9211	−8.3719	−9.1735	−9.8401	−9.1587
f_{23}	−7.8588	−8.6935	−9.4663	−9.2627	−9.8107
Function	$t = 0.5000$	$t = 0.7500$	$t = 1.0000$	$t = 1.2500$	$t = 1.5000$
f_1	1.8×10^{-4}	3.5×10^{-4}	5.7×10^{-4}	8.2×10^{-4}	0.0012
f_2	0.0041	0.0060	0.0081	0.0101	0.0120
f_{10}	0.0091	0.0136	0.0183	0.0227	9.1987
f_{11}	0.0274	0.0233	0.0161	0.0202	0.0121
f_{21}	−6.6272	−5.2845	−5.5189	−5.0095	−5.0578
f_{22}	−7.6829	−6.9698	−5.5194	−6.1831	−5.6476
f_{23}	−8.5037	−7.8622	−6.5713	−6.1300	−6.5364

7 Analysis of Fast and Classical Evolutionary Programming

It has been pointed out in Section 3 that Cauchy mutation has a higher probability of making long jumps than Gaussian mutation due to its long flat tails shown in Figure 1. In fact, the likelihood of a Cauchy mutation generating a larger jump than a Gaussian mutation can be estimated by a simple heuristic argument.

It is well known that if N_1 and N_2 are independent and identically distributed (i.i.d.) normal (Gaussian) random variates with density function

$$f_{Gaussian}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}},$$

then N_1/N_2 is Cauchy distributed with density function [16](pp.451)

$$f_{Cauchy}(x) = \frac{1}{\pi(1+x^2)}.$$

Given that Cauchy and Gaussian mutations follow the aforementioned distributions, Cauchy mutation will generate a larger jump than Gaussian mutation whenever $|N_1/N_2| > |N_1|$ (i.e., $|N_2| < 1.0$). Since the probability of a Gaussian random variate smaller than 1.0 is

$$\int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = 0.68,$$

Cauchy mutation is expected to generate a longer jump than Gaussian mutation with probability 0.68.

The expected length of Gaussian and Cauchy jumps can be calculated as follows:

$$E_{Gaussian}(x) = 2\int_{0}^{+\infty} x \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \frac{2}{\sqrt{2\pi}} = 0.80,$$

$$E_{Cauchy}(x) = 2\int_0^{+\infty} x\frac{1}{\pi(1+x^2)}dx = +\infty \quad \text{(i.e., does not exist)}.$$

It is obvious that Gaussian mutation is much more localized than Cauchy mutation.

Similar results can be obtained for Gaussian distribution with expectation $m > 0$ and variance $\sigma^2 > 1$ and Cauchy distribution with scale parameters $t > 1$. The question now is why larger jumps would be beneficial. Is this always true?

7.1 When Larger Jumps Are Beneficial

Sections 5.2 and 5.3.1 have explained qualitatively why larger jumps are good at dealing with plateaus and many local optima. This section shows analytically and empirically that this is true only when the global optimum is sufficiently far away from the current search point, i.e., when the distance between the current point and the global optimum is larger than the "step size" of the mutation.

Take the Gaussian mutation in CEP as an example, which uses the following distribution with expectation 0 (which implies the current search point is located at 0) and variance σ^2:

$$f_{G(0,\sigma^2)}(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}}, \quad -\infty < x < +\infty.$$

The probability of generating a point in the neighborhood of the global optimum x^* is given by:

$$P_{G(0,\sigma^2)}(|x - x^*| \leq \epsilon) = \int_{x^*-\epsilon}^{x^*+\epsilon} f_{G(0,\sigma^2)}(x)dx, \tag{5}$$

where $\epsilon > 0$ is the neighborhood size and σ is often regarded as the step size of the Gaussian mutation. Figure 5 illustrates the situation.

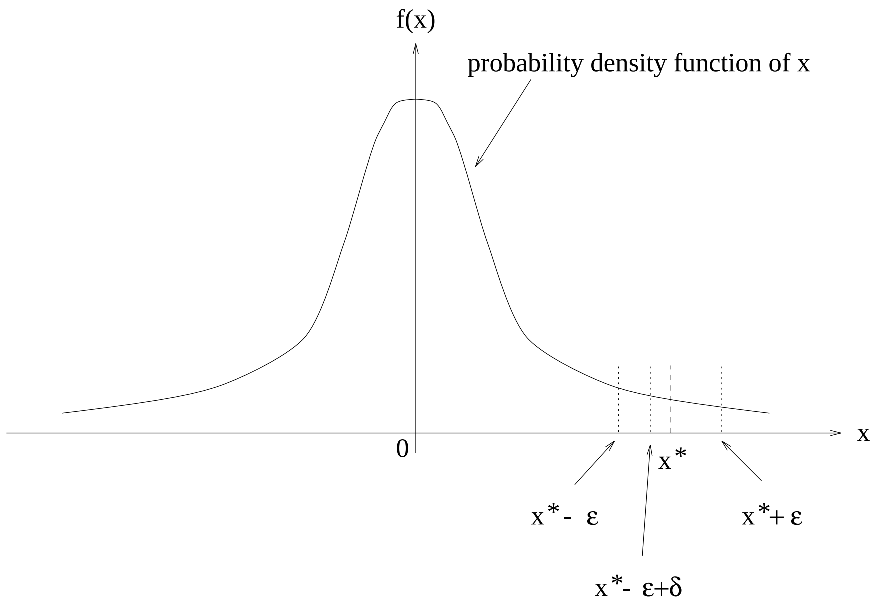


Figure 5: Evolutionary search as neighborhood search, where x^* is the global optimum and $\epsilon > 0$ is the neighborhood size. δ is a small positive number ($0 < \delta < 2\epsilon$).

The derivative $\frac{\partial}{\partial\sigma}P_{G(0,\sigma^2)}(|x - x^*| \leq \epsilon)$ can be used to evaluate the impact of σ on $P_{G(0,\sigma^2)}(|x - x^*| \leq \epsilon)$. According to the mean value theorem for definite integrals [17](pp.322), there exists a number δ ($0 < \delta < 2\epsilon$) such that

$$\int_{x^*-\epsilon}^{x^*+\epsilon} f_{G(0,\sigma^2)}(x)dx = 2\epsilon f_{G(0,\sigma^2)}(x^* - \epsilon + \delta).$$

Hence,

$$
\begin{aligned}
& \frac{\partial}{\partial \sigma} P_{G(0,\sigma^2)}(|x-x^*| \le \epsilon) \\
= \; & \frac{\partial}{\partial \sigma} \int_{x^*-\epsilon}^{x^*+\epsilon} f_{G(0,\sigma^2)}(x) dx \\
= \; & \frac{\partial}{\partial \sigma} \left(2\epsilon f_{G(0,\sigma^2)}(x^* - \epsilon + \delta) \right) \\
= \; & 2\epsilon \frac{\partial}{\partial \sigma} \left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x^*-\epsilon+\delta)^2}{2\sigma^2}} \right) \\
= \; & 2\epsilon \left(\frac{(x^*-\epsilon+\delta)^2}{\sigma^4\sqrt{2\pi}} e^{-\frac{(x^*-\epsilon+\delta)^2}{2\sigma^2}} - \frac{1}{\sigma^2\sqrt{2\pi}} e^{-\frac{(x^*-\epsilon+\delta)^2}{2\sigma^2}} \right) \\
= \; & \frac{2\epsilon}{\sigma^2\sqrt{2\pi}} e^{-\frac{(x^*-\epsilon+\delta)^2}{2\sigma^2}} \left(\frac{(x^*-\epsilon+\delta)^2}{\sigma^2} - 1 \right).
\end{aligned}
$$

It is apparent from the above equation that

$$\frac{\partial}{\partial \sigma} P_{G(0,\sigma^2)}(|x-x^*| \le \epsilon) > 0 \quad \text{if} \quad \sigma < |x^* - \epsilon + \delta|, \tag{6}$$

$$\frac{\partial}{\partial \sigma} P_{G(0,\sigma^2)}(|x-x^*| \le \epsilon) < 0 \quad \text{if} \quad \sigma > |x^* - \epsilon + \delta|. \tag{7}$$

That is, the larger σ is, the larger $P_{G(0,\sigma^2)}(|x-x^*| \le \epsilon)$ will be, if $\sigma < |x^* - \epsilon + \delta|$. However, if $\sigma > |x^* - \epsilon + \delta|$, the larger σ is, the smaller $P_{G(0,\sigma^2)}(|x-x^*| \le \epsilon)$ will be.

Similar analysis can be carried out for Cauchy mutation in FEP. Denote the Cauchy distribution defined by Eq.(3) as $f_{C(t)}(x)$. Then we have

$$
\begin{aligned}
& \frac{\partial}{\partial t} P_{C(t)}(|x-x^*| \le \epsilon) \\
= \; & \frac{\partial}{\partial t} \int_{x^*-\epsilon}^{x^*+\epsilon} f_{C(t)}(x) dx \\
= \; & \frac{\partial}{\partial t} \left(2\epsilon f_{C(t)}(x^* - \epsilon + \delta) \right) \\
= \; & \frac{2\epsilon}{\pi} \frac{\partial}{\partial t} \left(\frac{t}{t^2 + (x^*-\epsilon+\delta)^2} \right) \\
= \; & \frac{2\epsilon}{\pi} \left(\frac{1}{t^2 + (x^*-\epsilon+\delta)^2} - \frac{2t^2}{(t^2 + (x^*-\epsilon+\delta)^2)^2} \right) \\
= \; & \frac{2\epsilon}{\pi} \frac{(x^*-\epsilon+\delta)^2 - t^2}{(t^2 + (x^*-\epsilon+\delta)^2)^2},
\end{aligned}
$$

where δ $(0 < \delta < 2\epsilon)$ may not be the same as that in Eqs.(6) and (7). It is obvious that

$$\frac{\partial}{\partial t} P_{C(t)}(|x-x^*| \le \epsilon) > 0 \quad \text{if} \quad t < |x^* - \epsilon + \delta|, \tag{8}$$

$$\frac{\partial}{\partial t} P_{C(t)}(|x-x^*| \le \epsilon) < 0 \quad \text{if} \quad t > |x^* - \epsilon + \delta|. \tag{9}$$

That is, the larger t is, the larger $P_{C(t)}(|x-x^*| \le \epsilon)$ will be, if $t < |x^* - \epsilon + \delta|$. However, if $t > |x^* - \epsilon + \delta|$, the larger t is, the smaller $P_{C(t)}(|x-x^*| \le \epsilon)$ will be.

Since σ and t could be regarded as search step sizes for Gaussian and Cauchy mutations, the above analyses show that a large step size is beneficial (i.e., increases the probability of finding a near-optimal solution) only when the distance between the neighborhood of x^* and the current search point (at 0) is larger than the step size, or else a large step size may be detrimental to finding a near-optimal solution. The above analyses also show the rates of probability increase/decrease by deriving the explicit expressions for $\frac{\partial}{\partial\sigma}P_{G(0,\sigma^2)}(|x-x^*|\leq\epsilon)$ and $\frac{\partial}{\partial t}P_{C(t)}(|x-x^*|\leq\epsilon)$.

The analytical results explain why FEP achieved better results than CEP for most of the benchmark problems we tested, because the initial population was generated uniformly at random in a relatively large space and was far away from the global optimum on average. Cauchy mutation is more likely to generate larger jumps than Gaussian mutation and thus better in such cases. However, FEP would be less effective than CEP near the small neighborhood of the global optimum because Gaussian mutation's step size is smaller (smaller is better in this case). The experimental results on functions f_1 and f_2 illustrate such behavior clearly.

The analytical results also explain why FEP with a smaller t value for its Cauchy mutation would perform better whenever CEP outperforms FEP with $t = 1$. If CEP outperforms FEP with $t = 1$ for a problem, it implies that this FEP's search step size may be too large. In this case, using a Cauchy mutation with a smaller t is very likely to improve FEP's performance since it will have a smaller search step size. The experimental results presented in Table 6 match our theoretical prediction quite well.

7.2 Empirical Evidence

To validate the above analysis empirically, additional experiments were carried out. Function f_{21} (i.e., Shekel-5) was used here since it appears to pose some difficulties to FEP. First we made the search points closer to the global optimum by generating the initial population uniformly at random in the range of $2.5 \leq x_i \leq 5.5$ rather than $0 \leq x_i \leq 10$, and repeated our previous experiments. (The global optimum of f_{21} is at $x_i^* = 4$.) Such minor variation to the experiment is expected to improve the performance of both CEP and FEP since the initial search points are closer to the global optimum. Note that both Gaussian and Cauchy distributions have higher probabilities in generating points around 0 than those in generating points far away from 0.

The final experimental results averaged over 50 runs are given in Table 7. Figure 6 shows the results of CEP and FEP. It is quite clear that the performance of CEP improved much more than that of FEP since the smaller average distance between search points and the global optimum favors a small step size. The mean best of CEP improved significantly from -6.86 to -7.90, while that of FEP improved only from -5.52 to -5.62.

Table 7: Comparison of CEP's and FEP's final results on f_{21} when the initial population is generated uniformly at random in the range of $0 \leq x_i \leq 10$ and $2.5 \leq x_i \leq 5.5$. The results were averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation, and "Std Dev" stands for the standard deviation. The number of generations for each run was 100.

Initial Range	FEP		CEP		FEP−CEP
	Mean Best	Std Dev	Mean Best	Std Dev	t-test
$2.5 \leq x_i \leq 5.5$	-5.62	1.71	-7.90	2.85	$4.58^\dagger$
$0 \leq x_i \leq 10$	-5.57	1.54	-6.86	2.94	$2.94^\dagger$
t-test‡	-0.16		$-1.80^\dagger$		

†The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test. $^\ddagger\mathrm{FEP(CEP)}_{small}-\mathrm{FEP(CEP)}_{normal}$.

Then three more sets of experiments were conducted where the search space was expanded 10 times, 100 times, and 1000 times, i.e., the initial population was generated uniformly at random in the range of $0 \leq x_i \leq 100$, $0 \leq x_i \leq 1000$ and $0 \leq x_i \leq 10000$, and a_i's were multiplied by 10,

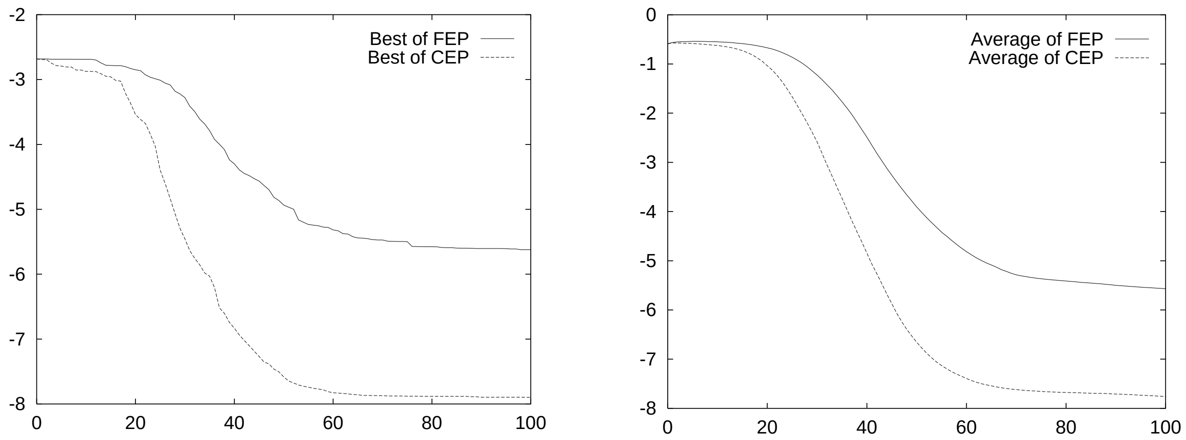


Figure 6: Comparison between CEP and FEP on f_{21} when the initial population is generated uniformly at random in the range of $2.5 \leq x_i \leq 5.5$. The solid lines indicate the results of FEP. The dotted lines indicate the results of CEP. The left figure shows the best result. The right figure shows the average result. Both were averaged over 50 runs. The horizontal axis indicates the number of generations. The vertical axis indicates the function value.

100 and 1000, respectively, making the average distance to the global optimum increasingly large. The enlarged search space is expected to make the problem more difficult and thus make CEP and FEP less efficient. The results of the same experiment averaged over 50 runs are shown in Table 8 and Figures 7, 8 and 9. It is interesting to note that the performance of FEP was less affected by the larger search space than CEP. When the search space was increased to $0 \leq x_i \leq 100$ and $0 \leq x_i \leq 1000$, the superiority of CEP over FEP on f_{21} disappeared. There was no statistically significant difference between CEP and FEP. When the search space was increased further to $0 \leq x_i \leq 10000$, FEP even outperformed CEP significantly. It is worth pointing out that a population size of 100 and the maximum number of generations of 100 are very small numbers for such a huge search space. The population might not have converged by the end of generation 100. However, this does not affect our conclusion. The experiments still show that Cauchy mutation performs much better than Gaussian mutation when the current search points are far away from the global optimum.

Even if a_i's were not multiplied by 10, 100, and 1000, similar results can still be obtained as long as the initial population was generated uniformly at random in the range of $0 \leq x_i \leq 100$, $0 \leq x_i \leq 1000$ and $0 \leq x_i \leq 10000$. Table 9 shows the results when a_i's were unchanged. The figures of this set of experiments are omitted to save some space. It is quite clear that a similar trend can be observed as the initial ranges increase.

It is worth reiterating that the only difference between the experiments in this section and the previous experiment in Section 5.3.2 is the range used to generate initial random populations. The empirical results match quite well with our analysis on the relationship between the step size and the distance to the global optimum. The results also indicate that FEP is less sensitive to initial conditions than CEP and thus more robust. In practice, the global optimum is usually unknown. There is little knowledge one can use to constrain the search space to a sufficiently small region. In such cases, FEP would be a better choice than CEP.

7.3 The Importance of Neighborhood Size

It is well known that finding an exact global optimum for a multimodal function is hard without prior knowledge about the function. It might take an infinite amount of time to find the global optimum for a global search algorithm such as CEP or FEP. In practice, one often has to sacrifice

Table 8: Comparison of CEP's and FEP's final results on f_{21} when the initial population is generated uniformly at random in the range of $0 \leq x_i \leq 10$, $0 \leq x_i \leq 100$, $0 \leq x_i \leq 1000$ and $0 \leq x_i \leq 10000$, and a_i's were multiplied by 10, 100 and 1000. The results were averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation, and "Std Dev" stands for the standard deviation. The number of generations for each run was 100.

Initial Range	FEP		CEP		FEP−CEP
	Mean Best	Std Dev	Mean Best	Std Dev	t-test
$0 \leq x_i \leq 10000$	−3.97	2.28	−2.60	2.43	$-4.02^{\dagger}$
$0 \leq x_i \leq 1000$	−5.00	2.96	−5.33	2.76	1.05
$0 \leq x_i \leq 100$	−5.80	3.21	−5.59	2.97	−0.40
$0 \leq x_i \leq 10$	−5.57	1.54	−6.86	2.94	$2.94^{\dagger}$
$\mathrm{F(C)EP}_{10000} - \mathrm{F(C)EP}_{1000}$	$2.73^{\dagger}$		$6.57^{\dagger}$		
$\mathrm{F(C)EP}_{1000} - \mathrm{F(C)EP}_{100}$	1.63		0.71		
$\mathrm{F(C)EP}_{100} - \mathrm{F(C)EP}_{10}$	−0.48		$2.10^{\dagger}$		

†The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

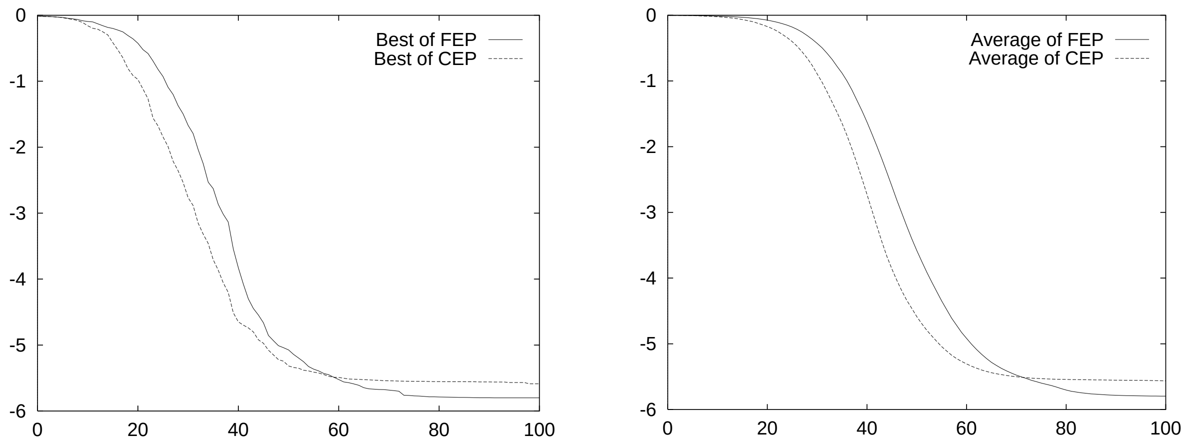


Figure 7: Comparison between CEP and FEP on f_{21} when the initial population is generated uniformly at random in the range of $0 \leq x_i \leq 100$ and a_i's were multiplied by 10. The solid lines indicate the results of FEP. The dotted lines indicate the results of CEP. The left figure shows the best result. The right figure shows the average result. Both were averaged over 50 runs. The horizontal axis indicates the number of generations. The vertical axis indicates the function value.

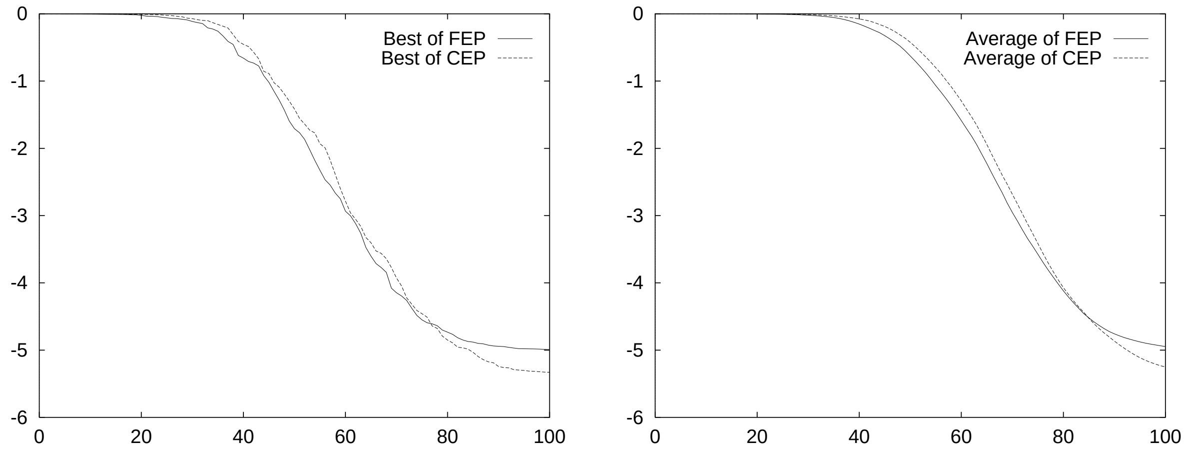


Figure 8: Comparison between CEP and FEP on f_{21} when the initial population is generated uniformly at random in the range of $0 \leq x_i \leq 1000$ and a_i's were multiplied by 100. The solid lines indicate the results of FEP. The dotted lines indicate the results of CEP. The left figure shows the best result. The right figure shows the average result. Both were averaged over 50 runs. The horizontal axis indicates the number of generations. The vertical axis indicates the function value.

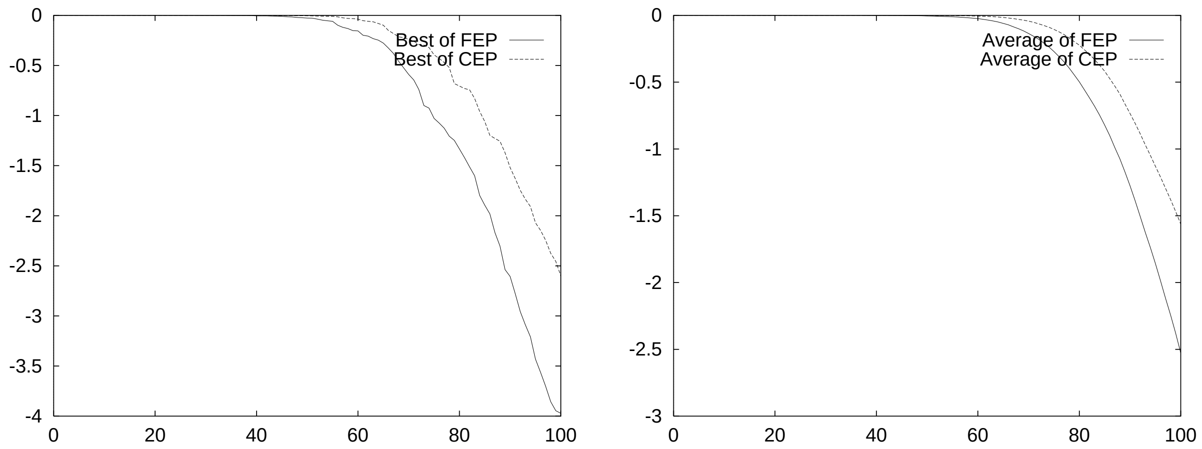


Figure 9: Comparison between CEP and FEP on f_{21} when the initial population is generated uniformly at random in the range of $0 \leq x_i \leq 10000$ and a_i's were multiplied by 1000. The solid lines indicate the results of FEP. The dotted lines indicate the results of CEP. The left figure shows the best result. The right figure shows the average result. Both were averaged over 50 runs. The horizontal axis indicates the number of generations. The vertical axis indicates the function value.

Table 9: Comparison of CEP's and FEP's final results on f_{21} when the initial population is generated uniformly at random in the range of $0 \leq x_i \leq 10$, $0 \leq x_i \leq 100$, $0 \leq x_i \leq 100$, $0 \leq x_i \leq 1000$ and $0 \leq x_i \leq 10000$. a_i's were unchanged. The results were averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation, and "Std Dev" stands for the standard deviation. The number of generations for each run was 100.

Initial Range	FEP		CEP		FEP−CEP
	Mean Best	Std Dev	Mean Best	Std Dev	t-test
$0 \leq x_i \leq 10000$	−4.12	1.45	−1.40	1.43	−10.44†
$0 \leq x_i \leq 1000$	−5.10	0.81	−5.15	1.66	0.19
$0 \leq x_i \leq 100$	−5.16	1.13	−5.61	2.31	1.21
$0 \leq x_i \leq 10$	−5.57	1.54	−6.86	2.94	2.94†
$\mathrm{F(C)EP}_{10000}-\mathrm{F(C)EP}_{1000}$	4.72†		12.46†		
$\mathrm{F(C)EP}_{1000}-\mathrm{F(C)EP}_{100}$	0.33		1.33		
$\mathrm{F(C)EP}_{100}-\mathrm{F(C)EP}_{10}$	1.48		2.24†		

†The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

discovering the global optimum in exchange for efficiency. A key issue that arises here is how much sacrifice one has to make in order to get a near-optimum in a reasonable amount of time. In other words, what is the relationship between the optimality of the solution and the time used to find the solution? This issue can be approached from the point of view of neighborhood size, i.e., ϵ in Eq.(5), since a smaller neighborhood size usually implies better optimality. It will be very useful if the impact of neighborhood size ϵ on the probability of *generating* a near-optimum in that neighborhood can be worked out. (The probability of finding a near-optimum would be the same as that of generating it when the elitism is used.) Although not an exact answer to the issue, the following analysis does provide some insights into such impact.

Similar to the analysis in Section 7.1, the following is true according to the mean value theorem for definite integrals [17](pp.322): for $0 < \delta < 2\epsilon$,

$$
\begin{aligned}
& \frac{\partial}{\partial \epsilon} P_{G(0,\sigma^2)}(|x - x^*| \leq \epsilon) \\
= \; & \frac{\partial}{\partial \epsilon} \int_{x^*-\epsilon}^{x^*+\epsilon} f_{G(0,\sigma^2)}(x) dx \\
= \; & \frac{\partial}{\partial \epsilon} \left(2\epsilon f_{G(0,\sigma^2)}(x^* - \epsilon + \delta) \right) \\
= \; & \frac{\partial}{\partial \epsilon} \left(2\epsilon \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x^*-\epsilon+\delta)^2}{2\sigma^2}} \right) \\
= \; & \frac{2}{\sigma\sqrt{2\pi}} \frac{\partial}{\partial \epsilon} \left(\epsilon e^{-\frac{(x^*-\epsilon+\delta)^2}{2\sigma^2}} \right) \\
= \; & \frac{2}{\sigma\sqrt{2\pi}} e^{-\frac{(x^*-\epsilon+\delta)^2}{2\sigma^2}} \left(1 + \frac{\epsilon}{\sigma^2} (x^* - \epsilon + \delta) \left(1 - \frac{\partial \delta}{\partial \epsilon} \right) \right).
\end{aligned}
$$

For the above equation, there exists a sufficiently small number $\epsilon_1 > 0$ such that for any $\epsilon \leq \epsilon_1$,

$$
\left| \frac{\epsilon}{\sigma^2} (x^* - \epsilon + \delta) \left(1 - \frac{\partial \delta}{\partial \epsilon} \right) \right| < 1.
$$

That is, for $0 < \epsilon \leq \epsilon_1$,

$$
\frac{\partial}{\partial \epsilon} P_{G(0,\sigma^2)}(|x - x^*| \leq \epsilon) > 0,
$$

which implies that the probability of generating a near-optimum increases as the neighborhood size increases in the vicinity of the optimum. The rate of such probability growth (i.e., $\frac{\partial}{\partial\epsilon}P_{G(0,\sigma^2)}(|x-x^*|\leq\epsilon)$) is governed by the term

$$e^{-\frac{(x^*-\epsilon+\delta)^2}{2\sigma^2}} = e^{-\frac{(x^*-(\epsilon-\delta))^2}{2\sigma^2}}.$$

That is, $\frac{\partial}{\partial\epsilon}P_{G(0,\sigma^2)}(|x-x^*|\leq\epsilon)$ grows exponentially faster as $\epsilon-\delta$ increases.

Similar analysis can be carried out for Cauchy mutation using its density function, i.e., Eq.(3). Let the density function be $C(1)$ when $t=1$. For $0<\delta<2\epsilon$,

$$\begin{aligned}
& \frac{\partial}{\partial\epsilon}P_{C(1)}(|x-x^*|\leq\epsilon) \\
= & \frac{\partial}{\partial\epsilon}\int_{x^*-\epsilon}^{x^*+\epsilon} f_{C(1)}(x)dx \\
= & \frac{\partial}{\partial\epsilon}\int_{x^*-\epsilon}^{x^*+\epsilon} \frac{1}{\pi(1+x^2)}dx \\
= & \frac{\partial}{\partial\epsilon}\left(\frac{1}{\pi}(\arctan(x^*+\epsilon)-\arctan(x^*-\epsilon))\right) \\
= & \frac{1}{\pi}\left(\frac{1}{1+(x^*+\epsilon)^2}+\frac{1}{1+(x^*-\epsilon)^2}\right) \\
> & 0.
\end{aligned}$$

Hence the probability of generating a near-optimum in the neighborhood always increases as the neighborhood size increases. While this conclusion is quite straightforward, it is interesting to note that the rate of increase in the probability differs significantly between Gaussian and Cauchy mutation since $\frac{\partial}{\partial\epsilon}P_{C(1)}(|x-x^*|\leq\epsilon) \gg \frac{\partial}{\partial\epsilon}P_{G(0,1)}(|x-x^*|\leq\epsilon)$.

8 An Improved Fast Evolutionary Programming

The previous analyses show the benefits of FEP and CEP in different situations. Generally, Cauchy mutation performs better when the current search point is far away from the global minimum, while Gaussian mutation is better at finding a local optimum in a good region. It would be ideal if Cauchy mutation is used when search points are far away from the global optimum and Gaussian mutation is adopted when search points are in the neighborhood of the global optimum. Unfortunately, the global optimum is usually unknown in practice, making the ideal switch from Cauchy to Gaussian mutation very difficult. Self-adaptive Gaussian mutation [7, 2, 8] is an excellent technique to partially address the problem. That is, the evolutionary algorithm itself will learn when to "switch" from one step size to another. However, there is room for further improvement to self-adaptive algorithms like CEP or even FEP.

This paper proposes an improved FEP (IFEP) based on *mixing* (rather than switching) different mutation operators. The idea is to mix different search biases of Cauchy and Gaussian mutations. The importance of search biases has been pointed out by some earlier studies [18](pp.375-376). The implementation of IFEP is very simple. It differs from FEP and CEP only in Step 3 of the algorithm described in Section 2. Instead of using Eq.(1) (for CEP) *or* Eq.(4) (for FEP) alone, IFEP generates two offspring from each parent, one by Cauchy mutation and the other by Gaussian. The better one is then chosen as the offspring. The rest of the algorithm is exactly the same as FEP and CEP. Chellapilla [19] has recently presented some more results on comparing different mutation operators in EP.

8.1 Experimental Studies

In order to carry out a fair comparison among IFEP, FEP, and CEP, the population size of IFEP was reduced to half of that of FEP or CEP in all the following experiments, since each individual in IFEP generates two offspring. However, reducing IFEP's population size by half actually disadvantages IFEP slightly because it does not double the time for any operators (such as selection) other than mutations. Nevertheless, such comparison offers a good and simple compromise.

IFEP was tested in the same experimental setup as before. For the sake of clarity and brevity, only some representative functions (out of 23) from each group were tested. Functions f_1 and f_2 are typical unimodal functions. Functions f_{10} and f_{11} are multimodal functions with many local minima. Functions f_{21}–f_{23} are multimodal functions with only a few local minima, and are particularly challenging to FEP. Table 10 summarizes the final results of IFEP in comparison with FEP and CEP. Figures 10 and 11 show the results of IFEP, FEP, and CEP.

Table 10: Comparison among IFEP, FEP, and CEP on functions $f_1, f_2, f_{10}, f_{11}, f_{21}, f_{22}, f_{23}$. All results have been averaged over 50 runs, where "Mean Best" indicates the mean best function values found in the last generation.

F	# of Gen's	IFEP Mean Best	FEP Mean Best	CEP Mean Best	IFEP−FEP t-test	IFEP−CEP t-test
f_1	1500	4.16×10^{-5}	5.72×10^{-4}	1.91×10^{-4}	$-28.06^{\dagger}$	$-2.39^{\dagger}$
f_2	2000	2.44×10^{-2}	7.60×10^{-2}	2.29×10^{-2}	$-51.61^{\dagger}$	$3.47^{\dagger}$
f_{10}	1500	4.83×10^{-3}	1.76×10^{-2}	8.79	$-48.54^{\dagger}$	$-21.26^{\dagger}$
f_{11}	2000	4.54×10^{-2}	2.49×10^{-2}	8.13×10^{-2}	$2.16^{\dagger}$	$-2.19^{\dagger}$
f_{21}	100	−6.46	−5.50	−6.43	$-2.19^{\dagger}$	$-5.46^{\dagger}$
f_{22}	100	−7.10	−5.73	−7.62	$-2.25^{\dagger}$	0.84
f_{23}	100	−7.80	−6.41	−8.86	$-2.14^{\dagger}$	$1.73^{\dagger}$

†The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

8.2 Discussions

It is very clear from Table 10 that IFEP has improved FEP's performance significantly for all test functions except for f_{11}. Even in the case of f_{11}, IFEP is better than FEP for 25 out of 50 runs. In other words, IFEP's performance is still rather close to FEP's and certainly better than CEP's (35 out of 50 runs) on f_{11}. These results show that IFEP continues to perform at least as well as FEP on multimodal functions with many minima, and also performs very well on unimodal functions and multimodal functions with only a few local minima with which FEP has difficulty handling. IFEP achieved performance similar to CEP's on these functions.

For the two unimodal functions where FEP is outperformed by CEP significantly, IFEP performs better than CEP on f_1, while worse than CEP on f_2. A closer look at the actual average solutions reveals that IFEP found much better solution than CEP on f_1 (roughly an order of magnitude smaller) while only performed slightly worse than CEP on f_2.

For the three Shekel functions f_{21}–f_{23}, the difference between IFEP and CEP is much smaller than that between FEP and CEP. IFEP has improved FEP's performance significantly on all three functions. It performs better than CEP on f_{21}, the same on f_{22}, and worse on f_{23}.

It is very encouraging that IFEP is capable of performing as well as or better than the better one of FEP and CEP for most test functions. This is achieved through a minimal change to the existing FEP and CEP. No prior knowledge or any other complicated operators were used. There

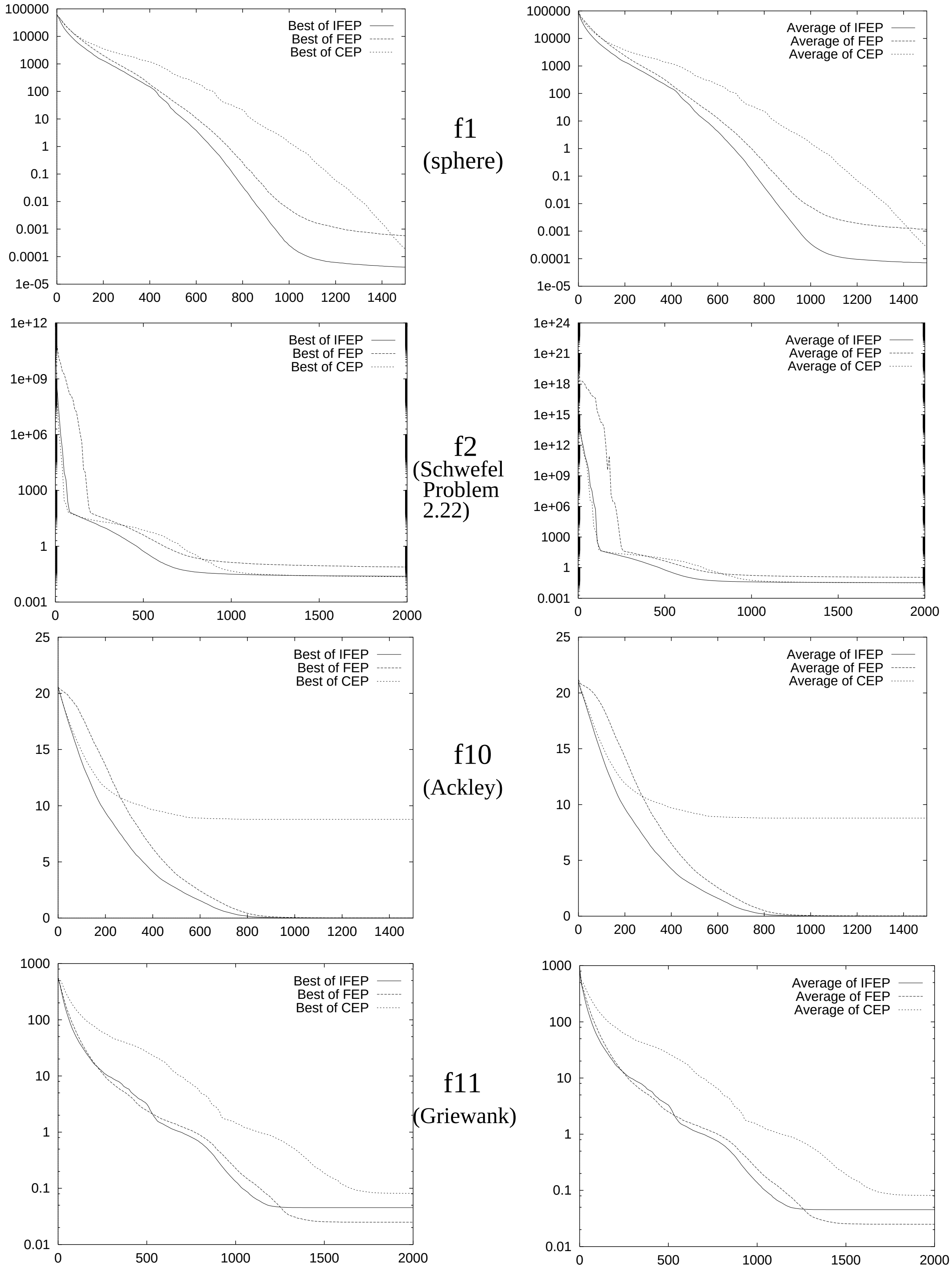


Figure 10: Comparison among IFEP, FEP and CEP on functions f_1, f_2, f_{10}, f_{11}. The vertical axis is the function value and the horizontal axis is the number of generations. The solid lines indicate the results of IFEP. The dashed lines indicate the results of FEP. The dotted lines indicate the results of CEP. The left figures show the best results. The right figures show the average results. All were averaged over 50 runs.

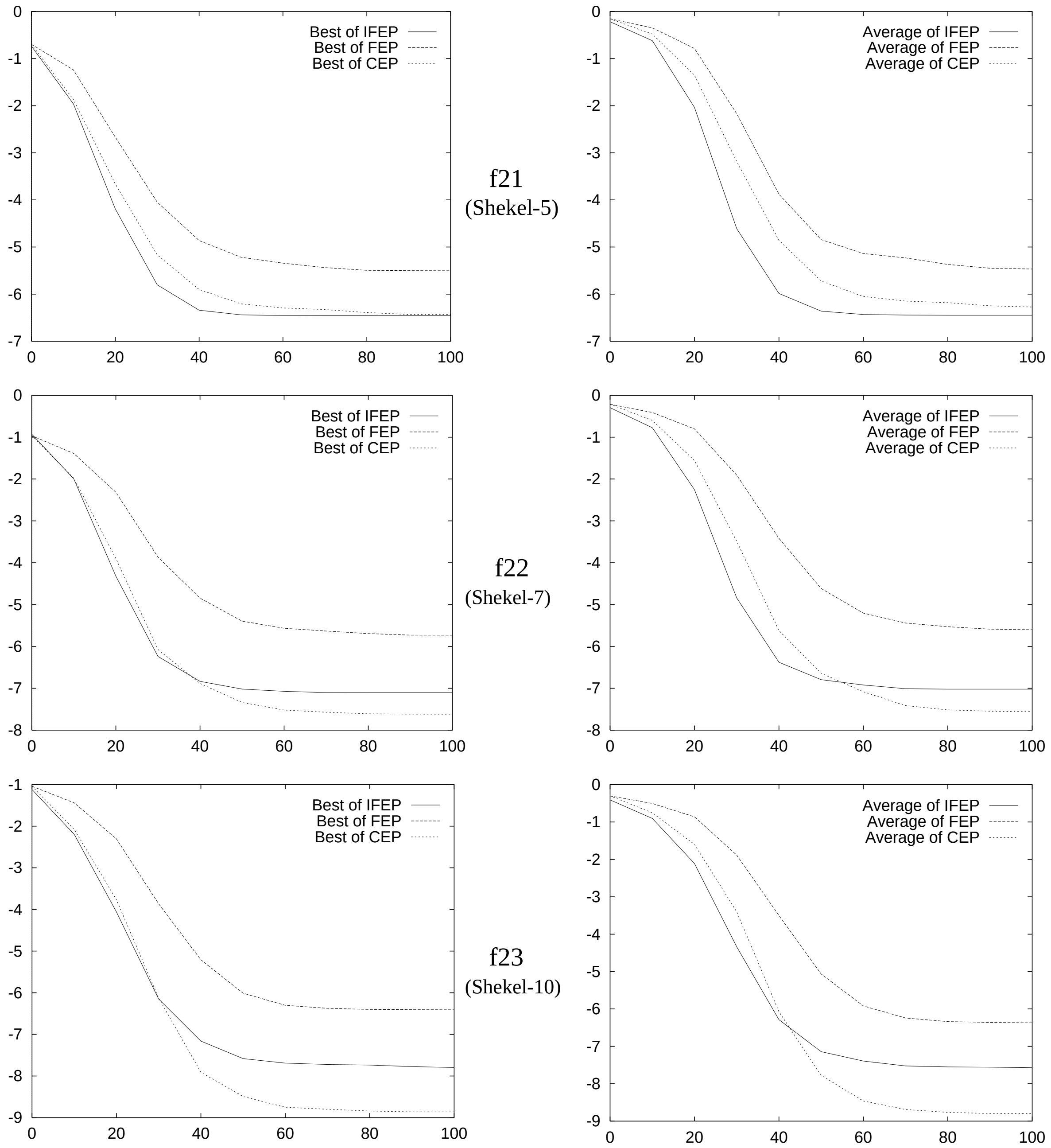


Figure 11: Comparison among IFEP, FEP and CEP on functions f_{21}–f_{23}. The vertical axis is the function value and the horizontal axis is the number of generations. The solid lines indicate the results of IFEP. The dashed lines indicate the results of FEP. The dotted lines indicate the results of CEP. The left figures show the best results. The right figures show the average results. All were averaged over 50 runs.

is no additional parameter used either. The superiority of IFEP also demonstrates the importance of mixing difference search biases (e.g., "step sizes") in a robust search algorithm.

The population size of IFEP used in the above experiments was only half of that of FEP and CEP. It is not unreasonable to expect even better results from IFEP if it uses the same population size as FEP's and CEP's. For (μ, λ) or $(\mu + \lambda)$ evolutionary algorithms where $\mu < \lambda$, it would be quite natural to use both Cauchy and Gaussian mutations since a parent needs to generate more than one offspring anyway.

It has been mentioned several times in this paper that Cauchy mutation performs better than Gaussian mutation because of its higher probability of making large jumps (i.e., having a larger expected search step size). However, according to our theoretical analysis, large search step sizes would be detrimental to search when the current search points are very close to the global optimum. Figures 12, 13, and 14 show the number of successful Cauchy mutations in a population in different generations. It is obvious that Cauchy mutation played a major role in the population in the early stages of evolution since the distance between the current search points and the global optimum was relatively large on average in the early stages. Hence Cauchy mutation performed better. However, as the evolution progressed, the distance became smaller and smaller. Large search step sizes produced by Cauchy mutation tended to produce worse offspring than those produced by Gaussian mutation. The decreasing number of successful Cauchy mutations in those figures illustrates this behavior.

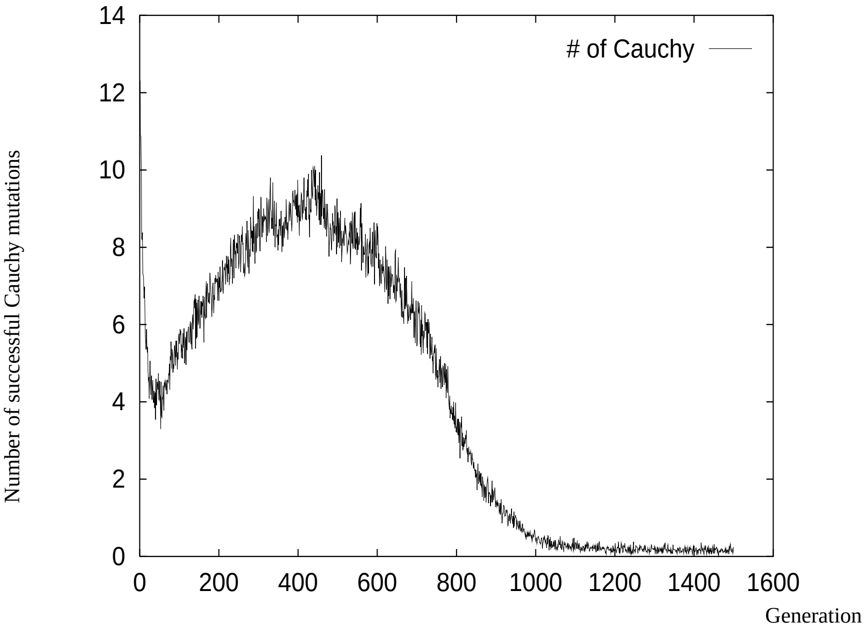


Figure 12: Number of successful Cauchy mutations in a population when IFEP is applied to function f_1. The vertical axis indicates the number of successful Cauchy mutations in a population and the horizontal axis indicates the number of generations. The results have been averaged over 50 runs.

9 Conclusions

This paper first proposes a fast evolutionary programming algorithm, FEP, and evaluates its performance on a number of benchmark problems. The experimental results show that FEP performs much better than CEP for multimodal functions with many local minima while being comparable to CEP in performance for unimodal and multimodal functions with only a few local minima. Since FEP and CEP differ only in their mutations, it is quite easy to apply FEP to real-world problems.

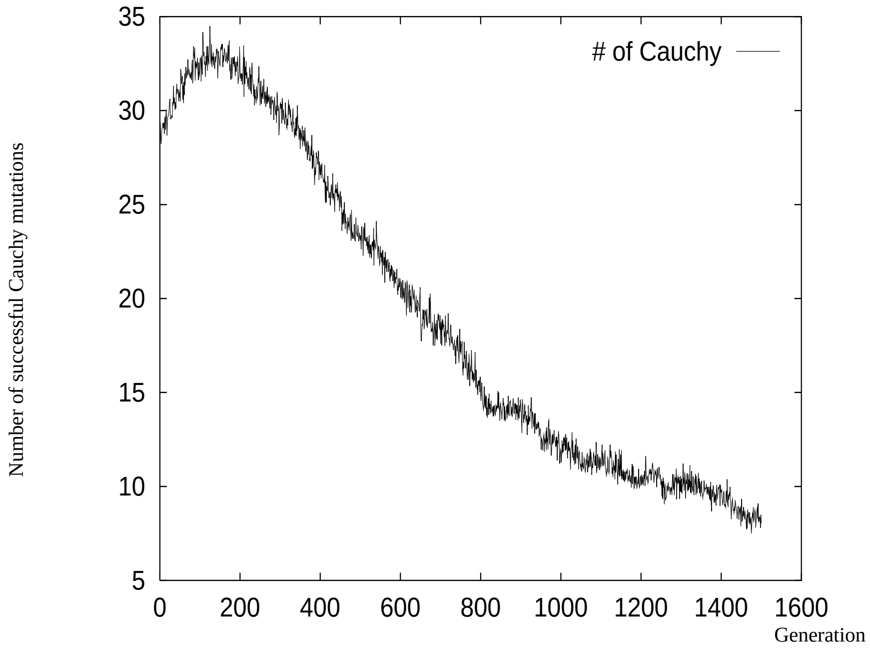


Figure 13: Number of successful Cauchy mutations in a population when IFEP is applied to function f_{10}. The vertical axis indicates the number of successful Cauchy mutations in a population and the horizontal axis indicates the number of generations. The results have been averaged over 50 runs.

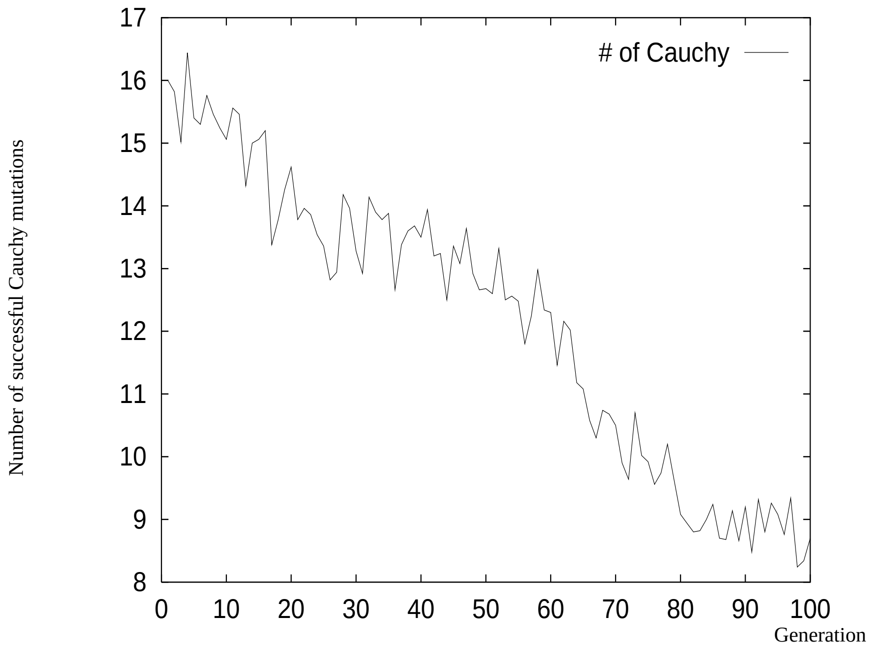


Figure 14: Number of successful Cauchy mutations in a population when IFEP is applied to function f_{21}. The vertical axis indicates the number of successful Cauchy mutations in a population and the horizontal axis indicates the number of generations. The results have been averaged over 50 runs.

No additional cost was introduced except for the difference in generating a Cauchy random number instead of a Gaussian random number.

The paper then analyzes FEP and CEP in depth in terms of search step size and neighborhood size, and explains why FEP performs better than CEP for most benchmark problems. The theoretical analysis is supported by the additional empirical evidence in which the range of initial x values was changed. The paper shows that FEP's long jumps increase the probability of finding a near-optimum when the distance between the current search point and the optimum is large, but decrease the probability when such distance is small. The paper also investigates the relationship between the neighborhood size and the probability of finding a near-optimum in this neighborhood. Some insights on evolutionary search and optimization in general have been gained from the above analyses.

The above analyses also led to an improved FEP (IFEP) which is very simple yet effective. IFEP uses the idea of mixing search biases to mix Cauchy and Gaussian mutations. Unlike some switching algorithms which have to decide when to switch between different mutations during search, IFEP does not need to make such decision and introduces no parameters. IFEP is robust, assumes no prior knowledge of the problem to be solved, and performs at least as well as the better one of FEP and CEP for most benchmark problems. Future work on IFEP includes the comparison of IFEP with other self-adaptive algorithms such as [20] and other evolutionary algorithms using Cauchy mutation [21].

The idea of FEP and IFEP can also be applied to other evolutionary algorithms to design faster optimization algorithms [22]. For $(\mu + \lambda)$ and (μ, λ) evolutionary algorithms where $\mu < \lambda$, IFES would be particularly attractive since a parent has to generate more than one offspring. It may be beneficial if different offspring are generated by different mutations [22].

10 Appendix: Benchmark Functions

10.1 Sphere Model

$$f_1(x) = \sum_{i=1}^{30} x_i^2$$

$$-100 \leq x_i \leq 100, \quad \min(f_1) = f_1(0, \ldots, 0) = 0$$

10.2 Schwefel's Problem 2.22

$$f_2(x) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|$$

$$-10 \leq x_i \leq 10, \quad \min(f_2) = f_2(0, \ldots, 0) = 0$$

10.3 Schwefel's Problem 1.2

$$f_3(x) = \sum_{i=1}^{30} \left(\sum_{j=1}^{i} x_j \right)^2$$

$$-100 \leq x_i \leq 100, \quad \min(f_3) = f_3(0, \ldots, 0) = 0$$

10.4 Schwefel's Problem 2.21

$$f_4(x) = \max_i \{|x_i|, 1 \le i \le 30\}$$

$$-100 \le x_i \le 100, \quad \min(f_4) = f_4(0, \ldots, 0) = 0$$

10.5 Generalized Rosenbrock's Function

$$f_5(x) = \sum_{i=1}^{29} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

$$-30 \le x_i \le 30, \quad \min(f_5) = f_5(1, \ldots, 1) = 0$$

10.6 Step Function

$$f_6(x) = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2$$

$$-100 \le x_i \le 100, \quad \min(f_6) = f_6(0, \ldots, 0) = 0$$

10.7 Quartic Function with Noise

$$f_7(x) = \sum_{i=1}^{30} i x_i^4 + random[0, 1)$$

$$-1.28 \le x_i \le 1.28, \quad \min(f_7) = f_7(0, \ldots, 0) = 0$$

10.8 Generalized Schwefel's Problem 2.26

$$f_8(x) = -\sum_{i=1}^{30} \left(x_i \sin\left(\sqrt{|x_i|}\right)\right)$$

$$-500 \le x_i \le 500, \quad \min(f_8) = f_8(420.9687, \ldots, 420.9687) = -12569.5$$

10.9 Generalized Rastrigin's Function

$$f_9(x) = \sum_{i=1}^{30} [x_i^2 - 10\cos(2\pi x_i) + 10)]$$

$$-5.12 \le x_i \le 5.12, \quad \min(f_9) = f_9(0, \ldots, 0) = 0$$

10.10 Ackley's Function

$$f_{10}(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30} \cos 2\pi x_i\right) + 20 + e$$

$$-32 \le x_i \le 32, \quad \min(f_{10}) = f_{10}(0, \ldots, 0) = 0$$

10.11 Generalized Griewank Function

$$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$-600 \le x_i \le 600, \quad \min(f_{11}) = f_{11}(0, \dots, 0) = 0$$

10.12 Generalized Penalized Functions

$$\begin{aligned} f_{12}(x) \;&=\; \frac{\pi}{30}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{29}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} \\ &\quad + \sum_{i=1}^{30} u(x_i, 10, 100, 4) \end{aligned}$$

$$-50 \le x_i \le 50, \quad \min(f_{12}) = f_{12}(1, \dots, 1) = 0$$

$$\begin{aligned} f_{13}(x) \;&=\; 0.1\left\{\sin^2(\pi 3x_1 + \sum_{i=1}^{29}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_{30})]\right\} \\ &\quad + \sum_{i=1}^{30} u(x_i, 5, 100, 4) \end{aligned}$$

$$-50 \le x_i \le 50, \quad \min(f_{13}) = f_{13}(1, \dots, 1) = 0$$

where

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

10.13 Shekel's Foxholes Function

$$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$$

$$-65.536 \le x_i \le 65.536, \quad \min(f_{14}) = f_{14}(-32, -32) \approx 1$$

where

$$(a_{ij}) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$$

10.14 Kowalik's Function

$$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$$

$$-5 \le x_i \le 5, \quad \min(f_{15}) \approx f_{15}(0.1928, 0.1908, 0.1231, 0.1358) \approx 0.0003075$$

Table 11: Kowalik's Function f_{15}

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

10.15 Six-hump Camel-Back Function

$$f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$
$$-5 \le x_i \le 5$$
$$x_{min} = (0.08983, -0.7126),\ (-0.08983, 0.7126)$$
$$\min(f_{16}) = -1.0316285$$

10.16 Branin Function

$$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$$
$$-5 \le x_1 \le 10, \quad 0 \le x_2 \le 15$$
$$x_{min} = (-3.142, 12.275),\ (3.142, 2.275),\ (9.425, 2.425)$$
$$\min(f_{17}) = 0.398$$

10.17 Goldstein-Price Function

$$\begin{aligned} f_{18}(x) &= [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ &\quad \times[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \end{aligned}$$
$$-2 \le x_i \le 2, \quad \min(f_{18}) = f_{18}(0, -1) = 3$$

10.18 Hartman's Family

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right]$$

with $n = 3, 6$ for $f_{19}(x)$ and $f_{20}(x)$, respectively, $0 \le x_j \le 1$. The coefficients are defined by Tables 12 and 13, respectively.

For $f_{19}(x)$ the global minimum is equal to -3.86 and it is reached at the point $(0.114, 0.556, 0.852)$. For $f_{20}(x)$ the global minimum is -3.32 at the point $(0.201, 0.150, 0.477, 0.275, 0.311, 0.657)$.

Table 12: Hartman Function f_{19}

i	$a_{ij}, j = 1,2,3$			c_i	$p_{ij}, j = 1,2,3$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.038150	0.5743	0.8828

Table 13: Hartman Function f_{20}

i	$a_{ij}, j = 1, \cdots, 6$						c_i	$p_{ij}, j = 1, \cdots, 6$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

10.19 Shekel's Family

$$f(x) = -\sum_{i=1}^{m} [(x - a_i)(x - a_i)^T + c_i]^{-1}$$

with $m = 5, 7$ and 10 for $f_{21}(x)$, $f_{22}(x)$ and $f_{23}(x)$, respectively, $0 \leq x_j \leq 10$.

Table 14: Shekel Functions f_{21}, f_{22}, f_{23}

i	$a_{ij}, j = 1, \cdots, 4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

These functions have 5, 7 and 10 local minima for $f_{21}(x)$, $f_{22}(x)$, and $f_{23}(x)$, respectively. $x_{local\_opt} \approx a_i$, $f(x_{local\_opt}) \approx 1/c_i$ for $1 \leq i \leq m$. The coefficients are defined by Table 14.

Acknowledgement — The authors are grateful to anonymous referees and Dr David Fogel for their constructive comments and criticism of earlier versions of this paper, and Drs Peter Angeline and Thomas Bäck for their insightful comments on self-adaptation in evolutionary algorithms.

References

[1] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution.*

New York, NY: John Wiley & Sons, 1966.

[2] D. B. Fogel, *System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling.*
Needham Heights, MA 02194: Ginn Press, 1991.

[3] D. B. Fogel, *Evolving Artificial Intelligence.*
PhD thesis, University of California, San Diego, CA, 1992.

[4] D. B. Fogel, "Applying evolutionary programming to selected traveling salesman problems," *Cybernetics and Systems*, vol. 24, pp. 27–36, 1993.

[5] X. Yao, "An overview of evolutionary computation," *Chinese Journal of Advanced Software Research (Allerton Press, Inc., New York, NY 10011)*, vol. 3, no. 1, pp. 12–29, 1996.

[6] D. B. Fogel, "An introduction to simulated evolutionary optimisation," *IEEE Trans. on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1994.

[7] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[8] D. B. Fogel, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence.*
New York, NY: IEEE Press, 1995.

[9] D. K. Gehlhaar and D. B. Fogel, "Tuning evolutionary programming for conformationally flexible molecular docking," in *Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming* (L. J. Fogel, P. J. Angeline, and T. Bäck, eds.), pp. 419–429, MIT Press, Cambridge, MA, 1996.

[10] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 2.
John Wiley & Sons, Inc., 2nd ed., 1971.

[11] X. Yao and Y. Liu, "Fast evolutionary programming," in *Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming* (L. J. Fogel, P. J. Angeline, and T. Bäck, eds.), (Cambridge, MA), pp. 451–460, The MIT Press, 1996.

[12] A. Törn and A. Žilinskas, *Global Optimisation.*
Berlin: Springer-Verlag, 1989.
Lecture Notes in Computer Science, Vol. 350.

[13] H.-P. Schwefel, *Evolution and Optimum Seeking.*
New York: John Wiley & Sons, 1995.

[14] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Tech. Rep. SFI-TR-95-02-010, Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA, July 1995.

[15] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[16] L. Devroye, *Non-Uniform Random Variate Generation.*
New York, NY 10010: Springer-Verlag, 1986.

[17] R. A. Hunt, *Calculus with Analytic Geometry.*
New York, NY 10022-5299: Harper & Row Publ., Inc., 1986.

[18] X. Yao, "Introduction," *Informatica (Special Issue on Evolutionary Computation)*, vol. 18, pp. 375–376, 1994.

[19] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 91–96, 1998.

[20] J. Born, "An evolution strategy with adaptation of the step sizes by a variance function," in *Parallel Problem Solving from Nature (PPSN) IV* (H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds.), vol. 1141 of *Lecture Notes in Computer Science*, (Berlin), pp. 388–397, Springer-Verlag, 1996.

[21] C. Kappler, "Are evolutionary algorithms improved by large mutations?," in *Parallel Problem Solving from Nature (PPSN) IV* (H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds.), vol. 1141 of *Lecture Notes in Computer Science*, (Berlin), pp. 346–355, Springer-Verlag, 1996.

[22] X. Yao and Y. Liu, "Fast evolution strategies," *Control and Cybernetics*, vol. 26, no. 3, pp. 467–496, 1997.