Evolutionary Dynamics of Evolutionary Programming in Noisy Environment

Yoshiyuki Matsumura
Kobe University
1-1, Rokkodai, Nada, Kobe
657-8501 JAPAN
matsumu@mi-2.mech.kobe-u.ac.jp

Kazuhiro Ohkura
Kobe University
1-1, Rokkodai, Nada, Kobe
657-8501 JAPAN
ohkura@mi-2.mech.kobe-u.ac.jp

Kanji Ueda
Kobe University
1-1, Rokkodai, Nada, Kobe
657-8501 JAPAN
ueda@mi-2.mech.kobe-u.ac.jp

Abstract- The effect of noise on evolutionary dynamics of Evolutionary Programming(EP) is empirically observed using three different statistical values. These are (1) the averaged best function values, (2) the average of strategy parameters and (3) Hotelling's T^2 of real values. The classical-EP(CEP), Fast-EP(FEP) and Robust-EP(REP) which stand for Fogel's EP, Yao and Liu's EP and our extended EP, respectively, are adopted for the observation. We show that REP performs more robustly than CEP and FEP for six noisy test functions.

1 Introduction

Noise is a common phenomenon in most real world problems. For example, in the information engineering field, any signal returned from the real world usually includes a significant amount of noise. In the field of Evolutionary Robotics, most simulation models are developed by taking noise into account in order to decrease the gap between simulated and real-world robots[11]. In such cases, Evolutionary Algorithms(EA) [2, 6, 8, 10, 19] work well even in the presence of noise. In fact, many EA approaches have been applied to optimizing noisy objective functions effectively[1, 3, 4, 9, 15]. Bäck and Hammel[3, 9] observed the evolutionary process of ES on the sphere function and it's convergence reliability on Rastrigin's function and Ackley's function. They demonstrated that noise does not influence the performance of ES as long as the noise level is small compared to the function value. Nissen and Propach[15] empirically investigated the performance of point-based methods, *e.g.*, Threshold Accepting and Pattern Search, and population-based methods, *e.g.*, ES and GA. Based on their experimental results, they concluded that, in the presence of noise, population-based methods were preferable to point-based methods. Beyer and Arnold[4] observed that noise reduces convergence velocity in both GA and ES. Angeline[1] compared the performance of the log-normal update rule and the Gaussian update rule in EP. He demonstrated that the Gaussian update rule outperforms the log-normal rule under some models of noise.

This paper attempts to characterise the evolutionary dynamics of EP applied to noisy objective functions by means of empirical experiments. The following statistical values are employed for this purpose: (1) the averaged best function values, (2) the average of strategy parameters and (3) Hotelling's T^2 of real values. The averaged best function values measures performance, the average of strategy parameters roughly defines the mutation step size and Hotelling's T^2 of real values show the balance between exploitation and exploration. Hotelling's T^2 can be applied to problems of any dimensionality. It was adopted for our experiments in the expectation that it would allow a characterisation of evolutionary dynamics that was independent of problem dimensionality. Of the many formulations of EP, Classical-EP, Fast-EP and Robust-EP are adopted for investigation.

The rest of this paper is organized as follows. Section 2 briefly introduces the computational procedures of CEP, FEP and REP. Section 3 shows the setup of computer experiments and the results. Finally, the conclusions are given in Section 4.

2 Computational Procedures of Evolutionary Programming

EP[6] was first proposed as an evolutionary approach to the design of finite state machines for solving prediction tasks. It has also been successfully applied to real-valued function optimization problems using normally distributed (Gaussian) mutation since the early 90's. Since then, there has been much discussion by researchers about which type of mutation performs best.

Yao and Liu[21] proposed the use of Cauchy mutation instead of Gaussian mutation and called their approach Fast-EP(FEP) to distinguish it from Fogel's Classical-EP(CEP). They found Cauchy mutation more effective for multimodal problems than Gaussian mutation, although no significant differences were observed for unimodal problems. Inspired by this, several types of combinations of Cauchy and Gaussian have been proposed[5, 18, 23, 24].

In these EPs, the mutation size of object variables is adjusted by it's own "self-adaptive" property, derived from strategy parameters. However, EP still suffers from premature convergence, *i.e.*, EP often converges before finding the global optimum, even when optimizing unimodal functions. Premature convergence is thought to be caused by a tendency for strategy parameters to fall too close to zero in early generations. The most popular scheme for avoiding this undesirable behavior is to constrain strategy parameters to be larger than a certain small positive value (lower bound). However, Liang, *et al.*[13] showed that the optimization performance of EP changes dramatically with different lower bound val-

Procedure O_{dup}

begin
$\tilde{\eta}_{i0}(j) = \eta_{i0}(j)$;
for p=1 to m **do**
  $\tilde{\eta}_{ip}(j) = \eta_{i(p-1)}(j)$;
end for
for p=0 to m **do**
  $\eta'_{kp}(j) = D(\tilde{\eta}_{ip}(j))$;
end for
end

Procedure O_{del}

begin
for p=0 to m-1 **do**
  $\tilde{\eta}_{ip}(j) = \eta_{i(p+1)}(j)$;
end for
$\tilde{\eta}_{im}(j) = \eta_L$
for p=0 to m **do**
  $\eta'_{kp}(j) = D(\tilde{\eta}_{ip}(j))$;
end for
end

Procedure O_{inv}

begin
$p = \mathbf{random}(1, \ldots, m)$;
$\tilde{\eta}_{i0}(j) = \eta_{ip}(j)$;
$\tilde{\eta}_{ip}(j) = \eta_{i0}(j)$;
$\eta'_{k0}(j) = D(\tilde{\eta}_{i0}(j))$;
$\eta'_{kp}(j) = D(\tilde{\eta}_{ip}(j))$;
end

Figure 1: Three new stochastic mutations for strategy parameters.

ues and the optimal lower bound is problem-dependent. This has been referred as *the lower bound problem*[16].

Robust-EP(REP)[14] was proposed to challenge this difficulty. Our motivation was simple: since selection itself fails to maintain diversity in strategy parameters so that the genetic search can continue, an alternative source for changing the parameters should be added. We selected *genetic drift*[12]. Genetic drift refers to stochastic changes in gene frequency, in contrast to the directional and deterministic changes caused by natural selection. We examined it's performance using static test functions and found that REP is free from the lower bound problem and more effective than conventional EP.

The computational steps of the EPs are based on notations in [24] and [14]. CEP, FEP and REP stand for Fogel's EP, Yao and Liu's EP and our extended EP, respectively.

2.1 Classical Evolutionary Programming

CEP is implemented as follows in this study.

1. Generate an initial population of μ individuals, and set $g = 1$. Each individual is taken as a pair of real-valued vectors ($\boldsymbol{x}_i$, $\boldsymbol{\eta}_i$), $\forall i \in \{1, \ldots, \mu\}$, where $\boldsymbol{x}_i$ and $\boldsymbol{\eta}_i$ are the i-th coordinate value in R^n and it's strategy parameters, larger than zero, respectively.

2. Evaluate the objective value for each individual ($\boldsymbol{x}_i$, $\boldsymbol{\eta}_i$), $\forall i \in \{1, \ldots, \mu\}$ of the population based on the objective function $f(\boldsymbol{x}_i)$.

3. Each parent ($\boldsymbol{x}_i$, $\boldsymbol{\eta}_i$), $i = 1, \ldots, \mu$, creates a single offspring. Offspring are calculated as follows: for $i = 1, \ldots, \mu$, $j = 1, \ldots, n$,

$$\eta'_i(j) = \eta_i(j) exp\{\tau' N(0,1) + \tau N_j(0,1)\} \quad (1)$$
$$x'_i(j) = x_i(j) + \eta'_i(j) N_j(0,1) \quad (2)$$

   where $x_i(j), x'_i(j), \eta_i(j)$ and $\eta'_i(j)$ denote the j-th component values of the vectors $\boldsymbol{x}_i$, $\boldsymbol{x}'_i$, $\boldsymbol{\eta}_i$ and $\boldsymbol{\eta}'_i$, respectively. $N(0,1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one. $N_j(0,1)$ 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 $\left(\sqrt{2n}\right)^{-1}$[2].

4. Calculate the objective value for each offspring ($\boldsymbol{x}_i$, $\boldsymbol{\eta}_i$), $\forall i \in \{1, \ldots, \mu\}$, according to $f(\boldsymbol{x}'_i)$.

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

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

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

2.2 Fast Evolutionary Programming

One of the extension forms of EP, called FEP (Fast EP), was proposed by Yao and Liu[21, 24]. It replaces Gaussian mutation with Cauchy mutation. Cauchy mutation uses the following Cauchy distribution function:

$$F_t(x) = 1/2 + (1/\pi)\arctan(x/t) \quad (3)$$

where $t = 1$. Empirical experiments showed the superiority of FEP over CEP especially on multimodal problems. The success of FEP is explained by there being a larger probability of escaping from local optima, due to the fatter tails of the Cauchy distribution[22].

2.3 Robust Evolutionary Programming

REP is designed to utilize the effect of genetic drift[12] by allowing selectively neutral mutations on strategy parameters, which enables a rapidly increasing or decreasing η_i irrespective of selection. REP follows the same procedures as FEP except for the following two points:

- New individual representation that holds redundant (i.e., inactive) strategy parameters. These parameters have no effect on the selection process.
- New stochastic mutation mechanisms for changing original (i.e., active) strategy parameters. These mutations replace, swap or copy active strategy parameters with inactive ones.

2.3.1 Individual Representation

An individual $\mathbf{X}_i$ is represented as follows, assuming that $i = 1, 2, \ldots, \mu$, $j = 1, 2, \ldots, n$, $k = 0, 1, \ldots, m$ and $\eta_{ik}(j) \in R^{+}$:

$$\mathbf{X}_i = [\boldsymbol{x}_i, (\boldsymbol{\eta}_{i0}, \ldots, \boldsymbol{\eta}_{ik}, \ldots, \boldsymbol{\eta}_{im})] \quad (4)$$

$$^t\boldsymbol{x}_i = (x_i(1), \ldots, x_i(j), \ldots, x_i(n)) \quad (5)$$

$$^t\boldsymbol{\eta}_{ik} = (\eta_{ik}(1), \ldots, \eta_{ik}(j), \ldots, \eta_{ik}(n)) \quad (6)$$

where $x_i(j)$ and $\eta_{ik}(j)$ denote the j-th component values of the vectors $\boldsymbol{x}_i$ and $\boldsymbol{\eta}_{ik}$, respectively. Notice that each $x_i(j)$ has $(m+1)$ strategy parameters.

2.3.2 Mutation Mechanisms for Strategy Parameters

Define D as same the mutation mechanism as Equation(1). $\boldsymbol{\eta}_{ik}$ is modified stochastically, according to the following new mutation operators (in Figure 1):

- O_{dup} shifts all of $\eta_{ik}(j)$ into the adjacent position of $(k+1)$ and removes $\eta_{im}(j)$ from the list. Then, modifies all $\boldsymbol{\eta}_{ik}$ with D.
- O_{del} discards $\eta_{i0}(j)$ and moves $\eta_{ik}(j)$ to the adjacent position of $(k-1)$. At the m-th position η_L is calculated as the smaller value either η_{max} or $\sum_{p=1}^{m-1} \eta_{ip}(j)$. Then, modifies all $\boldsymbol{\eta}_{ik}$ with D.
- O_{inv} swaps $\eta_{i0}(j)$ with one of $\eta_{ik}(j)$, $k = 1, \ldots, m$ and modifies $\eta_{i0}(j)$ and $\eta_{ik}(j)$ with D.

Notice that REP is equivalent to FEP when the probabilities of O_{dup}, O_{del} and O_{inv} are set at 1.0, 0.0 and 0.0, respectively.

3 Computer Simulations

3.1 Test Functions and Conditions

Following [3, 9], we calculate the noisy objective function $F(\boldsymbol{x}_i)$ at each generation as follows:

$$F(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) + \sigma_\delta N_i(0,1) \quad (7)$$

where the test functions $f(\boldsymbol{x}_i)$ are listed in Table 1. They are Sphere Model (f_1), Schwefel's Problem 2.22 (f_2), Step Function (f_3), Generalized Rastrigin's Function (f_4), Ackley's Function (f_5) and Generalized Griewank Function (f_6). All the test functions define 30 dimensional problems(n=30).

Table 1: Six test functions.

Functions ($n = 30$)	(Range)
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$(-100 \le x_i \le 100)$
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$(-10 \le x_i \le 10)$
$f_3(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	$(-100 \le x_i \le 100)$
$f_4(x) = \sum_{i=1}^{n} \{x_i^2 - 10\cos(2\pi x_i) + 10\}$	$(-5.12 \le x_i \le 5.12)$
$f_5(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e$	$(-32 \le x_i \le 32)$
$f_6(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$(-600 \le x_i \le 600)$

Functions f_1 to f_3 are unimodal functions and f_4 to f_6 are multimodal functions. $N_i(0,1)$ denotes a normally distributed n-dimensional random number with mean zero and standard deviation one which is generated anew for each value of i. We call σ_δ the noise level, and set it at 0.0, 0.001, 0.01, 0.1 and 1.0.

The experimental setup is based on Yao and Liu[21, 24]. All EP algorithms use the same sigma first[7] of the lognormal self-adaptation[17]. The population size μ is set at 100. The tournament size for selection **q** is 10. CEP, FEP and REP use the same initial populations, all simulations are independently repeated for 50 runs. The upper bound of strategy parameters η_{max} is set at 3.0. The lower bound of strategy parameters η_{min} is set at 10^{-6}. In REP, O_{dup}, O_{del} and O_{inv} are applied with the probabilities of 0.6, 0.3 and 0.1, respectively. The number of strategy parameters m for each variable is set at five. The main purpose of our computer simulations is to investigate the effect of noise on evolutionary dynamics. Thus, the parameters are not fully tuned.

3.2 The Averaged Best Function Values

The averaged best results of CEP, FEP and REP are shown in Figs.2 to Fig.4 for five noise levels, 0.0, 0.001, 0.01, 0.1 and 1.0. The effect of noise is observed for each test function.

In Fig.2(a) f_1, noise makes the convergence rate of CEP worse. CEP can not find function values less than 0. In Fig.3(a) f_1, noise makes the convergence rate of FEP better. But FEP can not find function values less than 0 before 1500 generations. In Fig.4(a) f_1, noise has no effect on the convergence rate of REP. REP can find function values less than 0 before 1500 generations.

In Fig.2(b) f_2, Fig.3(b) f_2 and Fig.4(b) f_2, the averaged best results of CEP, FEP and REP for f_2 show the same tendency as those for f_1.

In Fig.2(c) f_3 and Fig.3(c) f_3, CEP and FEP almost converge. On the contrary, in Fig.4(c) f_3, REP can find function values less than 0 before 1500 generations.

In Fig.2(d) f_4, Fig.3(d) f_4 and Fig.4(d) f_4, the averaged best results of CEP, FEP and REP for f_4 show the same tendency as that for f_3.

In Fig.2(e) f_5, CEP almost converges. In Fig.3(e) f_5 and Fig.4(e) f_5, FEP and REP can not find the solution when the noise level is 1.0. In the case of other noise levels, noise makes the convergence of FEP worse and REP can find function values less than 0 before 2500 generations.

In Fig.2(f) f_6, CEP can find function values less than 0 at 1200 generations when the noise level is 1.0. In the case of other noise level, CEP converges. In Fig.3(f) f_6, FEP can find function values less than 0 at 490 generations when the noise level is 1.0. In the case of other noise level, FEP converges. In Fig.4(f) f_6, REP can find function values less than 0 at 280 and at 1250 generations with noise levels of 1.0 and 0.01 respectively. At the other noise levels, REP converges.

From these results it can be said that noise has a greater effect on the evolutionary dynamics of CEP and FEP than it does on REP.

3.3 The Average of Strategy Parameters

The average of $\eta_i(1)$ for f_1 is shown in Figs. 5 to 7 for the five noise levels of 0.0, 0.001, 0.01, 0.1 and 1.0. The average of other strategy parameters $\eta_i(j)(j = 2, \ldots, n)$ shows the same tendency as $\eta_i(1)$.

In the case of CEP and FEP, the average of $\eta_i(1)$ gradually decreases ($\sim 10^{-6}$) for all noise levels. In contrast, with REP, the average of $\eta_i(1)$ rapidly increases and decreases for all noise levels. From these results it can be said that noise has little effect on the evolutionary dynamics of CEP, FEP and REP with respect to strategy parameters.

However, the reason why the average of strategy parameters for CEP and FEP approached 10^{-6} was because the lower bound was set at 10^{-6}. This means that CEP and FEP still suffer from premature convergence in the case of noisy environment and the mutation size is not sufficiently adjusted by it's own self-adaptive property. On the other hand, the average of strategy parameters for REP oscillates rapidly between 10^{-1} and minute values approaching 10^{-6}. Since strategy parameters roughly determine the mutation size, this means that during the course of the run, individuals repeatedly switch between making large and small movements through the search space.

3.4 Hotelling's T^2 of Real Values

Hotelling's T^2 is a statistical measure of the multivariate distance of each observation from the center of a data set. In an evolutionary population, individuals located far from the center of the population can be said to be engaged in exploration, while those located close to the center are engaged in exploitation. Hotelling's T^2 is useful because it gives a measure of the ratio of population individuals engaged in local search (exploitation) to those engaged in global search (exploration).

It is calculated as follows:

$$\boldsymbol{T}^2 = \mu(\mu-1)(\bar{\boldsymbol{x}}-\boldsymbol{\delta})'\boldsymbol{S}^{-1}(\bar{\boldsymbol{x}}-\boldsymbol{\delta}) \quad (8)$$

$$\bar{\boldsymbol{x}} = \sum_{i=1}^{\mu} \frac{\boldsymbol{x}_i}{\mu} \quad (9)$$

$$\boldsymbol{S} = \sum_{i=1}^{\mu} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})' \quad (10)$$

where μ is the number of data points. Assume that $\boldsymbol{x}_i$ follow $N(\boldsymbol{\delta}, \boldsymbol{\sigma_\zeta})$. $N(\boldsymbol{\delta}, \boldsymbol{\sigma_\zeta})$ is an N-dimensional normal distribution in which $\boldsymbol{\delta}$ and $\boldsymbol{\sigma_\zeta}$ are the average vector and the covariance, respectively. If $\boldsymbol{x}_i$ has the probability density function $f(\boldsymbol{x})$, $\boldsymbol{\delta}$ is calculated as follows:

$$\begin{aligned} f(\boldsymbol{x}) &= (2\pi)^{-n/2}|\boldsymbol{\sigma_\zeta}|^{-1/2} \quad (11) \\ &\quad \times exp[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\delta})'\boldsymbol{\sigma_\zeta}^{-1}(\boldsymbol{x}-\boldsymbol{\delta})] \\ E(\boldsymbol{x}_i) &= \boldsymbol{\delta} \quad (12) \end{aligned}$$

Hotelling's T^2 of $\boldsymbol{x}_i$ for f_1 is shown in Figs.8 to Figs.10 for the five noise levels, 0.0, 0.001, 0.01, 0.1 and 1.0.

In Fig.8(a), when the noise level is 0.0, the number of individuals which are more than 60 of Hotelling's T^2 value gradually decreases from 10 to 5 as the runs progress. On the contrary, in Fig.8(b), (c), (d) and (e), when noise is present, the number of individuals which are more than 60 of Hotelling's T^2 value gradually increases from 0 to 20 as the runs progress.

In Fig.9(a), when the noise level is 0.0, there is little change in the balance between exploration (i,e. Hotelling's T^2 value is more than 80) and exploitation (i,e. Hotelling's T^2 value is less than 20) over generations. On the contrary, in Fig.9(b), (c), (d) and (e), there is a random change in the balance between exploitation and exploration over generations.

In Fig.10(a), (b), (c), (d) and (e), there is no clear difference even when noise exists. The distribution changes in such a way that ultimately 20% of individuals are located very far from the center of the population, while 40% of individuals are located very close to the center. REP gradually shifted toward a strategy composed of fine exploitation and broad exploration.

3.5 Summary

To summarize our computer simulations the three following points are stated.

- While REP shows a noise-level-independent performance, both CEP and FEP show brittle performance facing different noise levels.

- During the evolutionary runs, the average values of strategy parameters for both CEP and FEP decrease gradually, implying that they might still suffer from the premature convergence. However, in REP, the average values of strategy parameters change dramatically throughout the run, showing that individuals are continually changing between making large and small movements through the search space.

- In the presence of noise, both CEP and FEP showed that the balance of exploitation and exploration was changing randomly over generations. On the other hand, REP gradually shifted toward a strategy composed of fine exploitation and broad exploration.

4 Conclusions

In this paper, we applied EP to noisy objective functions. Computer simulations were conducted on six test functions in order to investigate the effect of noise on evolutionary dynamics of EP. The results suggested that REP performed more robustly in the presence of noise than CEP and FEP. In the future, we would like to apply EP to artificial neural networks in real-world problems (i.e, Evolutionary Artificial Neural Networks (EANN)[20, 25] in Evolutionary Robotics).

Acknowledgments

The authors acknowledge partial financial support from the “Methodology of Emergent Synthesis” project (96P00702) and Grant-in-Aid for Scientific Research (11450060) from JSPS (the Japan Society for the Promotion of Science). The first author is also supported by JSPS Research Fellowship for Young Scientists(04999).

Bibliography

[1] P. J. Angeline (1996), “ The Effects of Noise on Self-Adaptive Evolutionary Optimization”, *Proc. of the 5th Annual Conference on Evolutionary Programming*, pp.433-439, MIT Press.

[2] T. Bäck (1996), *Evolutionary Algorithms in Theory and Practice*, Oxford University Press.

[3] T. Bäck and U. Hammel (1994), “Evolution Strategies Applied to Perturbed Objective Functions”, *Proc. of the First IEEE Conference on Evolutionary Computation*, pp.40-45, IEEE Press.

[4] H.-G. Beyer and D. Arnold (1999), ”Fitness Noise and Localization Errors of the Optimum in General Quadratic Fitness Models”, *Proc. of Genetic and Evolutionary Computation Conference (GECCO’99)*, pp. 817-824, Morgan Kaufmann.

[5] K. Chellapilla (1998), “Combining Mutation Operators in Evolutionary Programming”, *IEEE Transactions on Evolutionary Computation*, Vol. 2, No. 3, pp.91-96.

[6] D. B. Fogel (1995), *Evolutionary Computation Toward a New Philosophy of Machine Intelligence*, IEEE Press.

[7] D. K. Gehlhaar and D. B. Fogel (1996), “Tuning Evolutionary Programming for Conformationally Flexible Molecular Docking”, *Proc. of the 5th Annual Conference on Evolutionary Programming*, pp. 419-429, MIT Press.

[8] D. Goldberg (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

[9] U. Hammel and T. Bäck (1994), ”Evolution Strategies on Noisy Functions: How to Improve Convergence Properties”, *Proc. of Parallel Problem solving from Nature - PPSN III*, pp.418-427, Springer-Verlag.

[10] J. H. Holland (1992), *Adaptation in Natural and Artificial Systems*, MIT Press edition.

[11] N. Jakobi, P.Husbands and I. Harvey (1995), “Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics”, *Third European Conference on Artificial Life(ECAL95)*, pp.704-720, Springer-verlge.

[12] M. Kimura (1983), *The Neutral Theory of Molecular Evolution*, Cambridge University Press.

[13] K.-H. Liang, X. Yao, Y. Liu, C. Newton and D. Hoffman (1998), “An Experimental Investigation of Self-adaptation in Evolutionary Programming”, *Proc. of the 7th Annual Conference on Evolutionary Programming*, pp.291-300, Springer-Verlag.

[14] Y. Matsumura, K. Ohkura and K. Ueda (1999), ”Evolutionary Programming with Non-coding Segments for Real-valued Function Optimization”, *Proc. of 1999 IEEE International Conference on Systems, Man and Cybernetics (SMC’99)*, Vol.4, pp.242-247.

[15] V. Nissen and J. Propach (1998), “Optimization with Noisy Function”, *Proc. of Parallel Problem solving from Nature - PPSN V*, pp.159-168, Springer-Verlag.

[16] K. Ohkura, Y. Matsumura and K. Ueda (2001), “Robust Evolution Strategies”, *Applied Intelligence*, Kluwer Academic Publishers, in printing.

[17] N. Saravanan and D. B. Fogel (1994), “Learning Strategy Parameters in Evolutionary Programming: An Empirical Study”, *Proc. of the 3rd Annual Conference on Evolutionary Programming*, pp.269-280, World Scientific Publishing.

[18] N. Saravanan and D. B. Fogel (1997), “Multi-Operator Evolutionary Programming: A Preliminary Study on Function Optimization”, *Proc. of the 6th Annual Conference on Evolutionary Programming*, Lecture Notes in Computer Science Vol. 1213, pp.215-221, Springer-Verlag.

[19] H.-P. Schwefel (1995), *Evolution and Optimum Seeking*, John Wiley & Sons.

[20] X. Yao (1993), “A Review of Evolutionary Artificial Neural Networks”, *International Journal of Intelligent Systems* Vol.8, pp. 539-567.

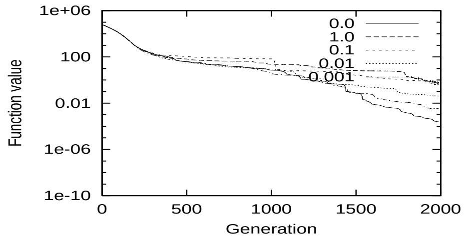
[21] X. Yao and Y. Liu (1996), “ Fast Evolutionary Programming”, *Proc. of the 5th Annual Conference on Evolutionary Programming*, pp.451-460, MIT Press.

[22] X. Yao, G. Lin and Y. Liu (1997), “ An Analysis of Evolutionary Algorithms Based on Neighborhood and Step Sizes”, *Proc. of the 6th Annual Conference on Evolutionary Programming*, Lecture Notes in Computer Science Vol. 1213, pp.297-307, Springer-Verlag.

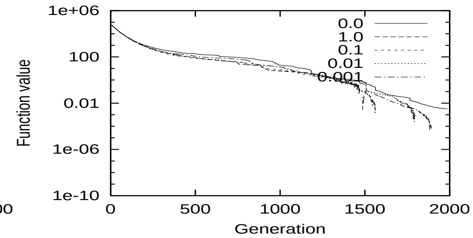
[23] X. Yao and Y. Liu (1998), “ Scaling up Evolutionary Programming Algorithms”, *Proc. of the 7th Annual Conference on Evolutionary Programming*, pp.103-112, Springer-Verlag.

[24] X. Yao, Y. Liu and G. Lin (1999), “ Evolutionary Programming Made Faster”, *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp.82-102.

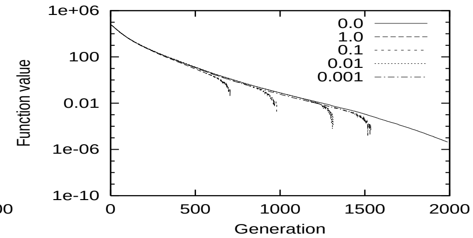
[25] X. Yao (1999), “Evolving artificial neural networks,” *Proc. of the IEEE*, 87(9), pp.1423-1447.



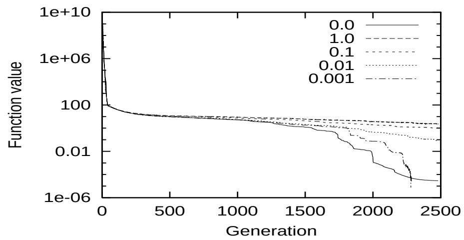
(a) f_1



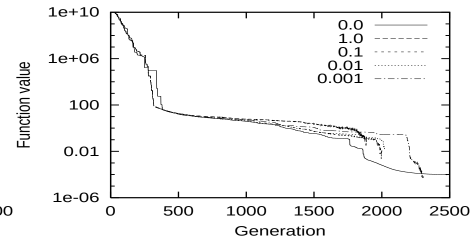
(a) f_1



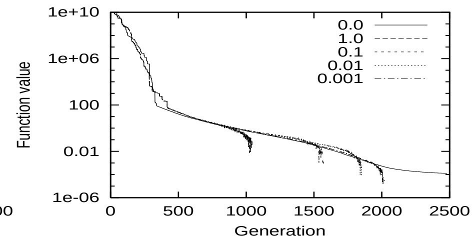
(a) f_1



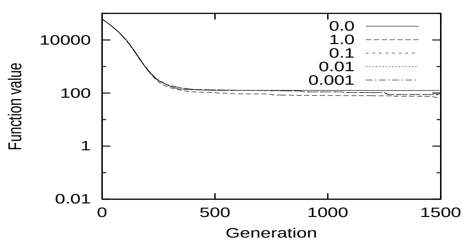
(b) f_2



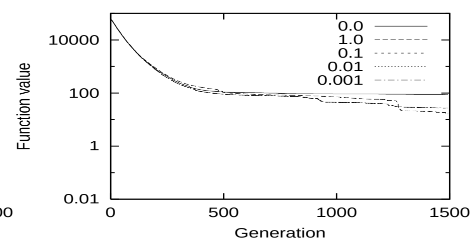
(b) f_2



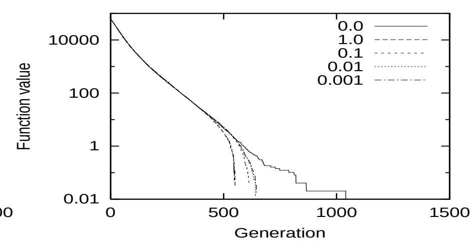
(b) f_2



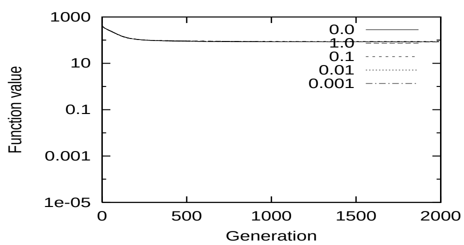
(c) f_3



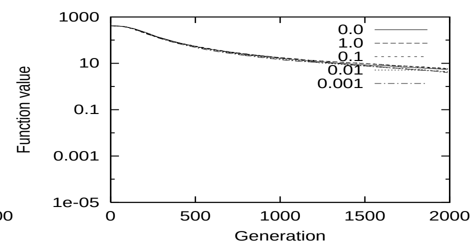
(c) f_3



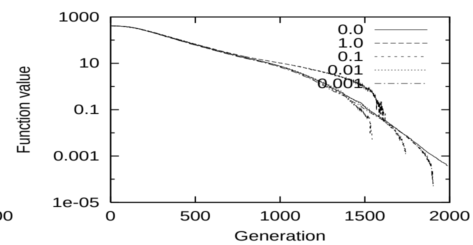
(c) f_3



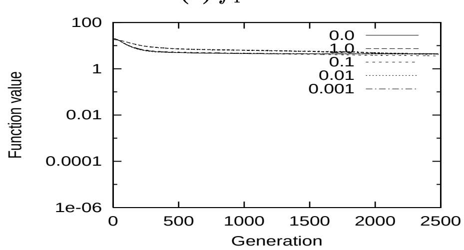
(d) f_4



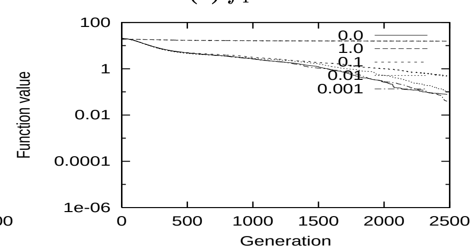
(d) f_4



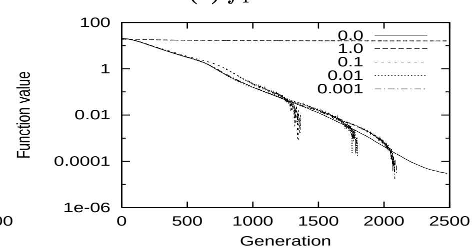
(d) f_4



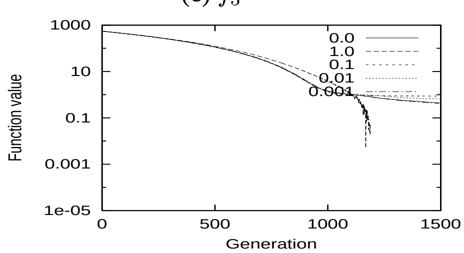
(e) f_5



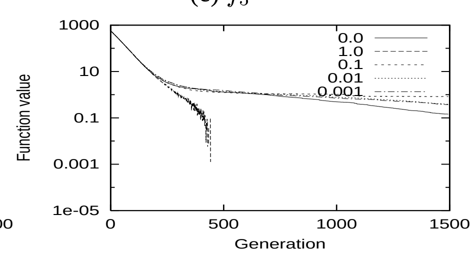
(e) f_5



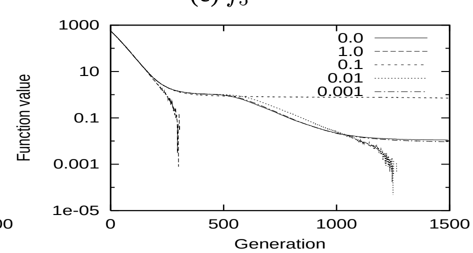
(e) f_5



(f) f_6



(f) f_6



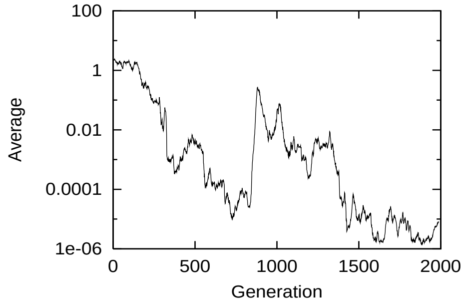
(f) f_6

Figure 2: CEP

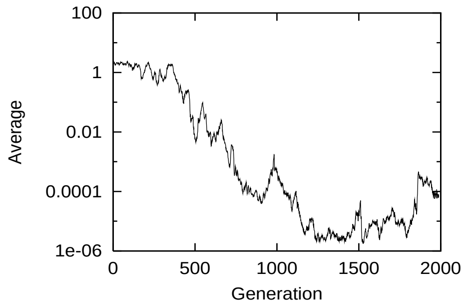
Figure 3: FEP

Figure 4: REP

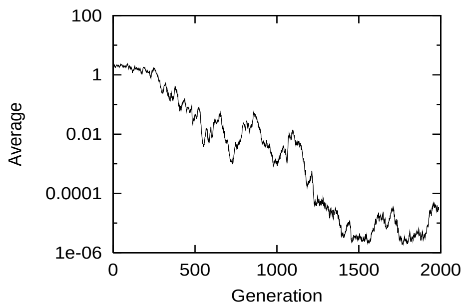
Figs. 2 to 4: The averaged best results when the noise level(σ_δ) is 0.0, 0.001, 0.01, 0.1 and 1.0.



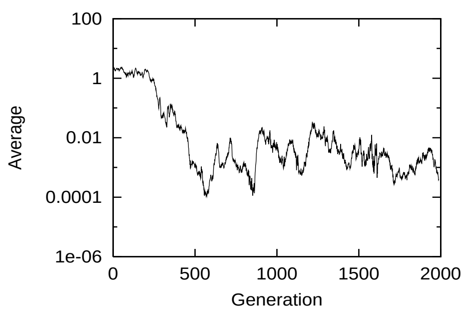
(a) σ_δ=0.0



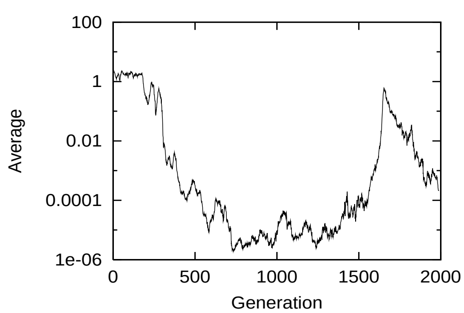
(b) σ_δ=0.001



(c) σ_δ=0.01

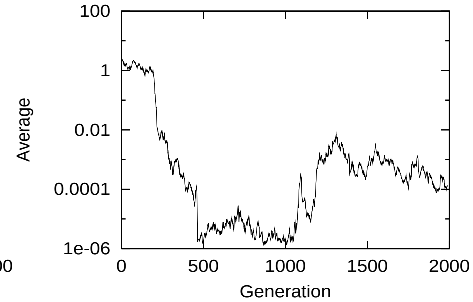


(d) σ_δ=0.1

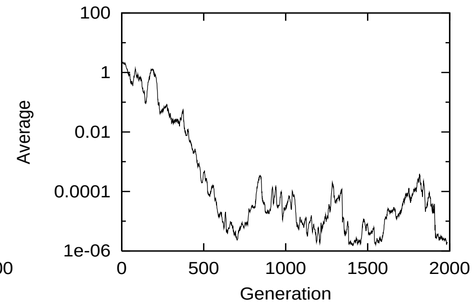


(e) σ_δ=1.0

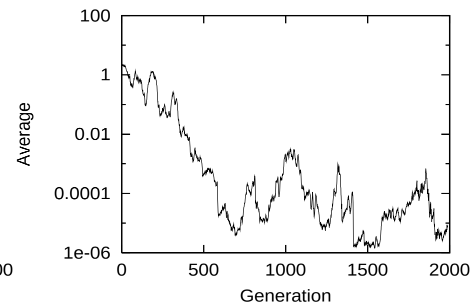
Figure 5: CEP



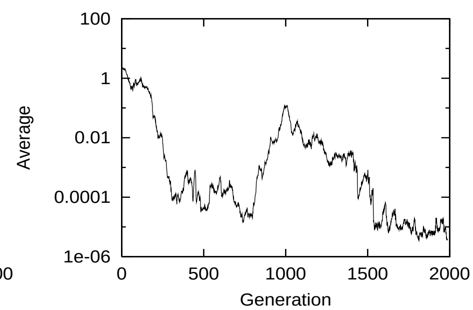
(a) σ_δ=0.0



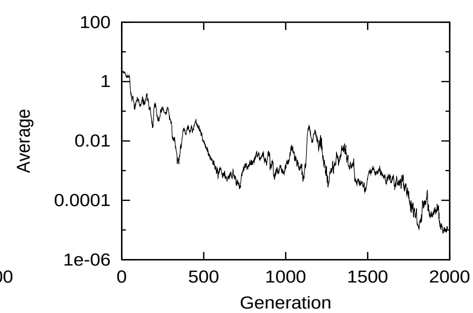
(b) σ_δ=0.001



(c) σ_δ=0.01

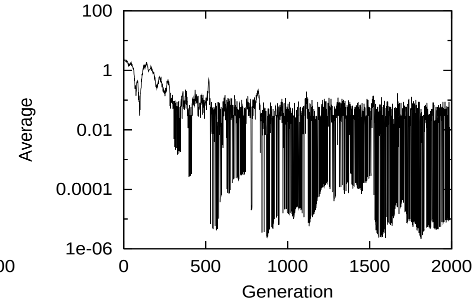


(d) σ_δ=0.1

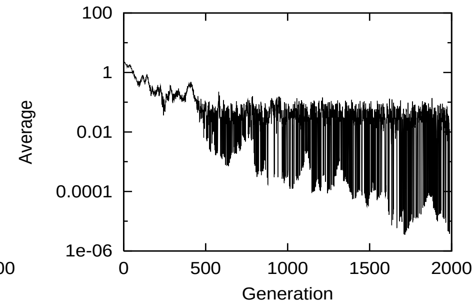


(e) σ_δ=1.0

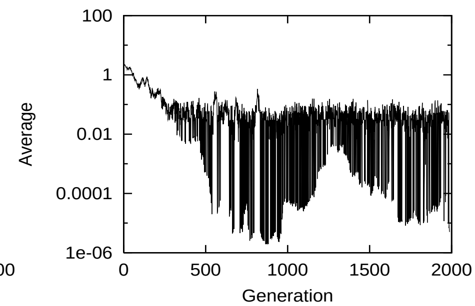
Figure 6: FEP



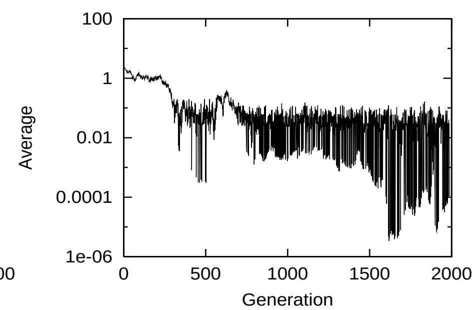
(a) σ_δ=0.0



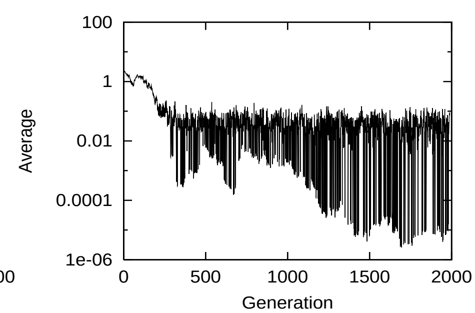
(b) σ_δ=0.001



(c) σ_δ=0.01



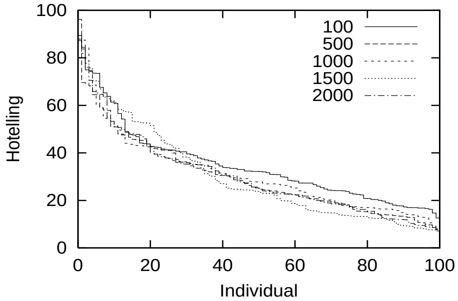
(d) σ_δ=0.1



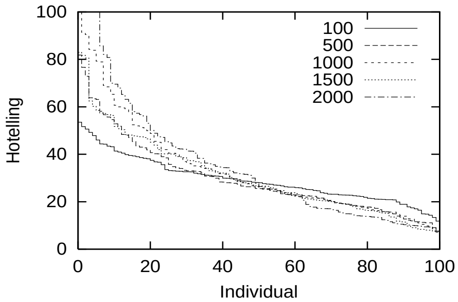
(e) σ_δ=1.0

Figure 7: REP

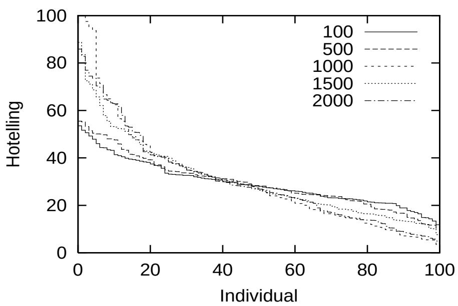
Figs. 5 to 7: The average of $\eta_i(1)$ for f_1.



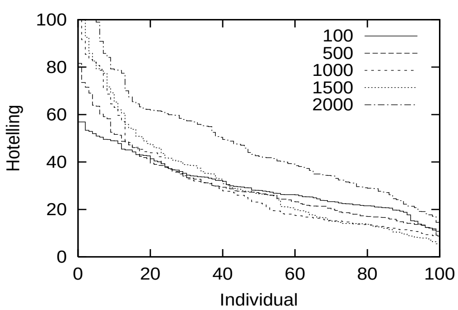
(a) σ_δ=0.0



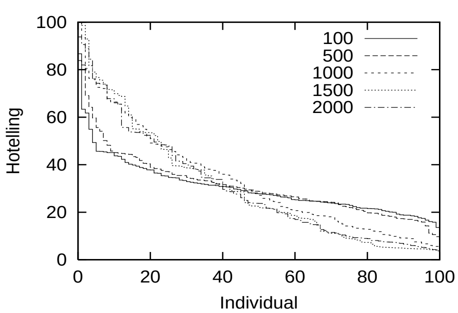
(b) σ_δ=0.001



(c) σ_δ=0.01

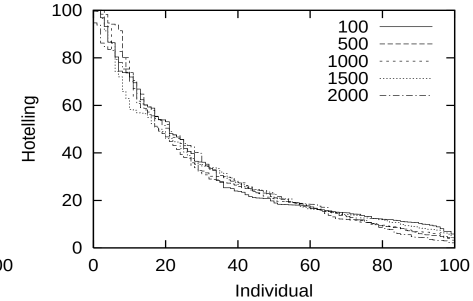


(d) σ_δ=0.1

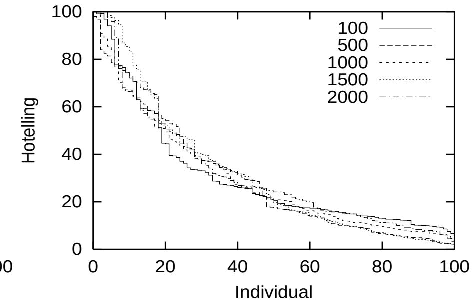


(e) σ_δ=1.0

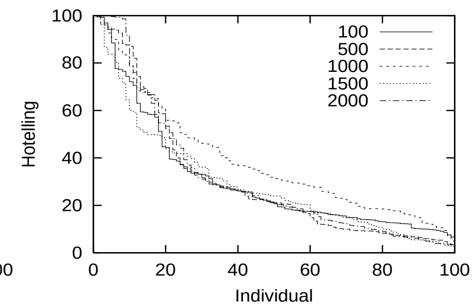
Figure 8: CEP



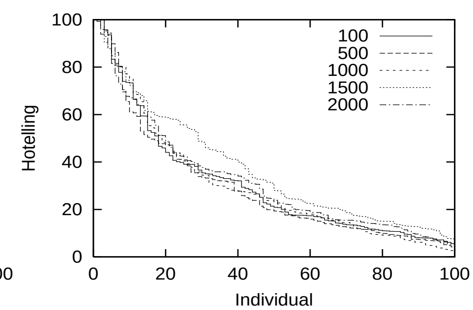
(a) σ_δ=0.0



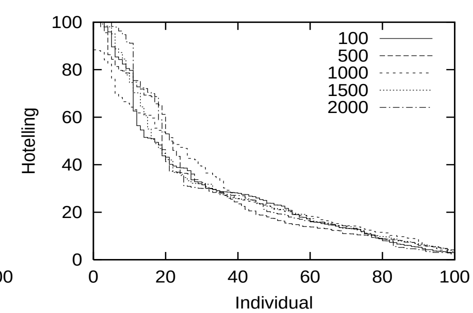
(b) σ_δ=0.001



(c) σ_δ=0.01

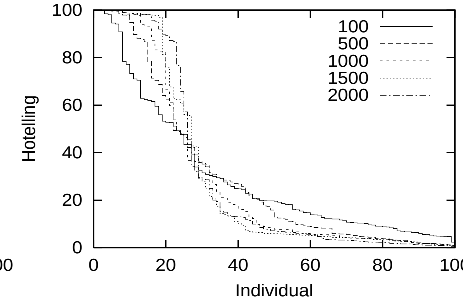


(d) σ_δ=0.1

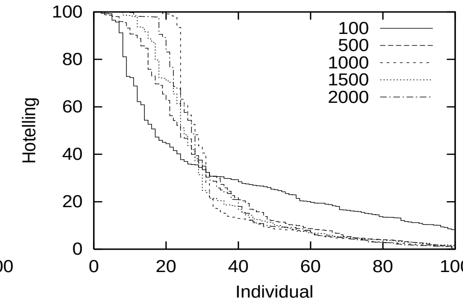


(e) σ_δ=1.0

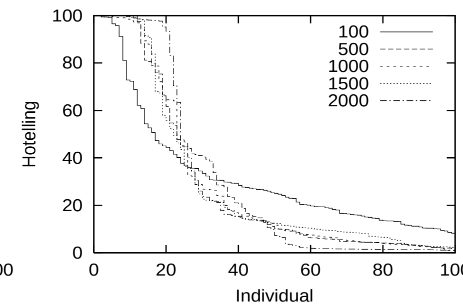
Figure 9: FEP



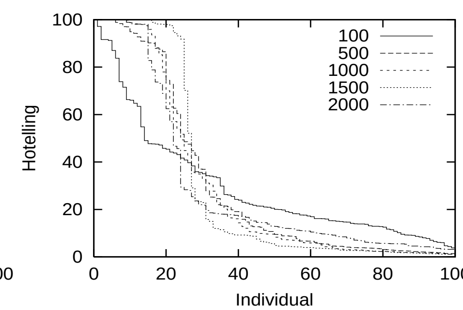
(a) σ_δ=0.0



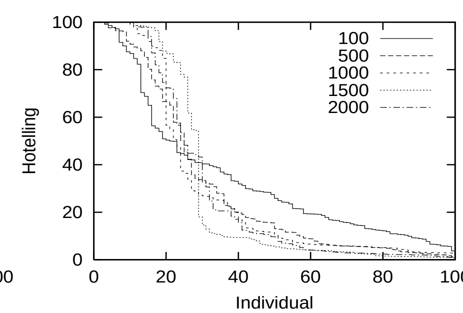
(b) σ_δ=0.001



(c) σ_δ=0.01



(d) σ_δ=0.1



(e) σ_δ=1.0

Figure 10: REP

Figs. 8 to 10: Hotelling's T^2 of $\boldsymbol{x}_i$ for f_1 when the generation is 100, 500, 1000, 1500 and 2000.