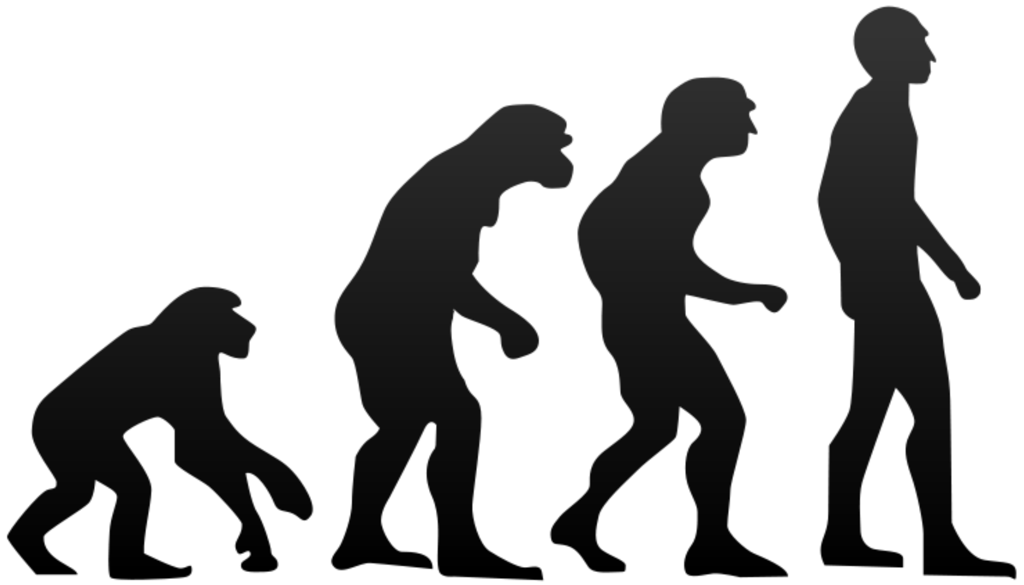
Metaheuristics II



Outline of This Lecture

- Global Optimization by Mutation-Based EAs
  - Classical Evolutionary Programming (CEP)
  - Fast Evolutionary Programming (FEP)
- Test problems and Comparison
- Search Step Size and Search Bias
- Summary

A Simpler EA

- Let’s look at an EA with mutation only.
- It’s basically mutation + selection.

Global Optimization by Mutation-Based EAs

1. Generate the initial population of μ individuals, and set $k = 1$. Each individual is a real-valued vector $\boldsymbol{x_i}$.

2. Evaluate the fitness of each individual.

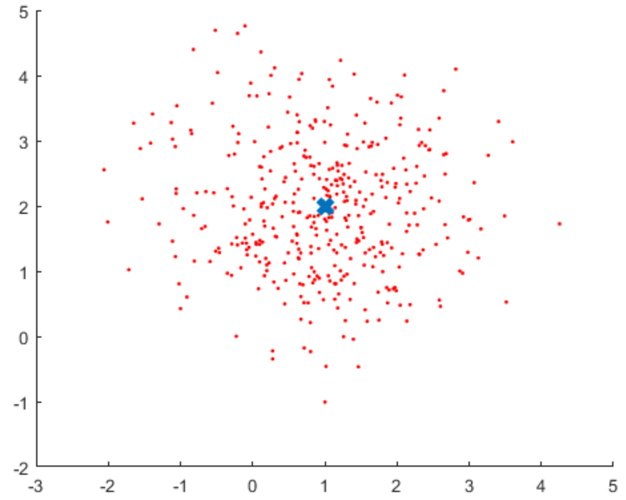
Global Optimization by Mutation-Based EAs

3. Each individual creates a single offspring: for $j = 1, \ldots, n$,

$$\boldsymbol{x}_{\boldsymbol{i}}^{'}(j) = \boldsymbol{x}_{\boldsymbol{i}}(j) + N_j(0,1)$$

where $\boldsymbol{x}_{\boldsymbol{i}}(j)$denotes the j^{th} component of the vectors $\boldsymbol{x}_{\boldsymbol{i}}$.

$N(0,1)$ 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 newly generated for each value of j.



Global Optimization by Mutation-Based EAs

4. Calculate the fitness of each offspring.
5. For each individual, q opponents are chosen randomly from all the parents and offspring with an equal probability. For each comparison, if the individual's fitness is no greater than the opponent's, it receives a "win."
6. Select the μ best individuals (from 2μ) that have the most wins to be the next generation.
7. Stop if the stopping criterion is satisfied; otherwise, $k = k + 1$ and go to Step 3.

Why $N(0,1)$?

- The standard deviation of the normal distribution determines the search step size of the mutation. It is a crucial parameter.
- Unfortunately, the optimal search step size is problem-dependent.
- Even for a single problem, different search stages require different search step sizes.
- Self-adaptation can be used to get around this problem partially.

What Do Mutation and Self-Adaptation Do

Classical EP

Fast EP

Test Problems

Optimization by Classical EP (CEP)

EP = Evolutionary Programming

1. Generate the initial population of μ individuals and set $k = 1$. Each individual is taken as a pair of real-valued vectors, $(\boldsymbol{x_i}, \boldsymbol{\eta_i}), \forall i \in \{1, \dots, \mu\}$. Unfortunately, the optimal search step size is problem-dependent.

2. Evaluate the fitness score for each individual $(\boldsymbol{x_i}, \boldsymbol{\eta_i}), \forall i \in \{1, \dots, \mu\}$, of the population based on the objective function, $f(\boldsymbol{x_i})$. Self-adaptation can be used to get around this problem partially.

Function Optimization by Classical EP (CEP)

3. Each parent $(\boldsymbol{x_i}, \boldsymbol{\eta_i}), \forall i \in \{1, \dots, \mu\}$ creates a single offspring $\left(\boldsymbol{x'_i}, \boldsymbol{\eta'_i}\right)$ by:

For $j \in \{1, \dots, n\}$:

$$x'_i(j) = x_i(j) + \eta_i(j) N_j(0,1) \qquad (1)$$

$$\eta'_i(j) = \eta_i(j) \exp\left(\tau' N(0,1) + \tau N_j(0,1)\right) \qquad (2)$$

where $x'_i(j)$, $x'_i(j)$, $\eta_i(j)$ and $\eta'_i(j)$ denote the j^{th}component 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 factor τ and τ'has commonly set to $1/(\sqrt{2\sqrt{n}})$ and $\frac{1}{\sqrt{2n}}$.

4. Calculate the fitness of each offspring, $\left(\boldsymbol{x'_i}, \boldsymbol{\eta'_i}\right), \forall i \in \{1, \dots, \mu\}$.

Function Optimization by Classical EP (CEP)

5. Conduct pairwise comparison over the union of parents $(\boldsymbol{x_i}, \boldsymbol{\eta_i})$ and offspring $(\boldsymbol{x'_i}, \boldsymbol{\eta'_i}), \forall i \in \{1, \dots, \mu\}$. For each individual, q opponents are chosen randomly from all the parents and offspring with an equal probability. For each comparison, if the individual's fitness is no greater than the opponent's, it receives a ``win“.

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

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

Fast EP

- The idea comes from fast simulated annealing.
- Use a Cauchy, instead of Gaussian, random number in Eq. (1) to generate a new offspring. That is,

$$\boldsymbol{x}_i'(j) = \boldsymbol{x_i}(j) + \boldsymbol{\eta_i}(j)\delta_j$$

where δ_j is a Cauchy random number variable with the scale parameter $t = 1$, and is generated anew for each value of j.

- Everything else, including Eq. (2), are kept unchanged in order to evaluate the impact of Cauchy random numbers.

1-d Cauchy Distribution

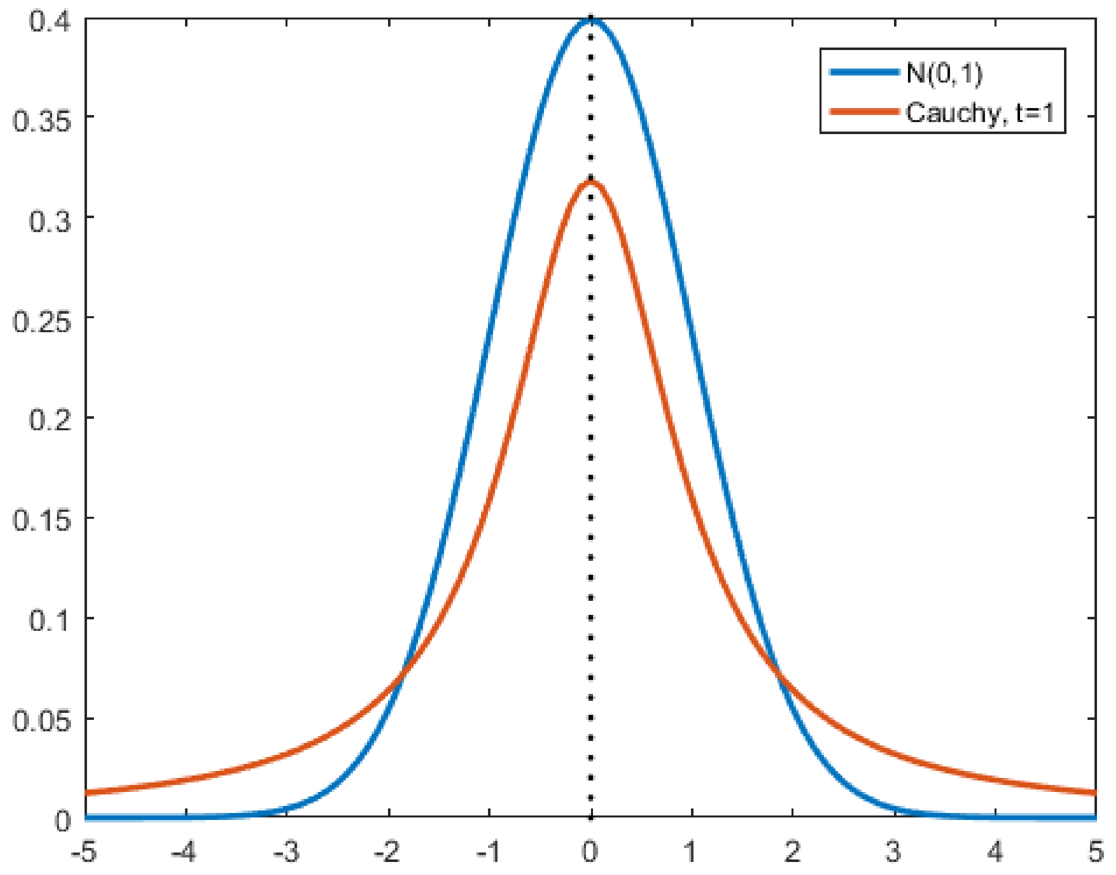
Its density function is

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

where t>0 is a scale parameter. The corresponding distribution function is

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

Gaussian and Cauchy Density Functions

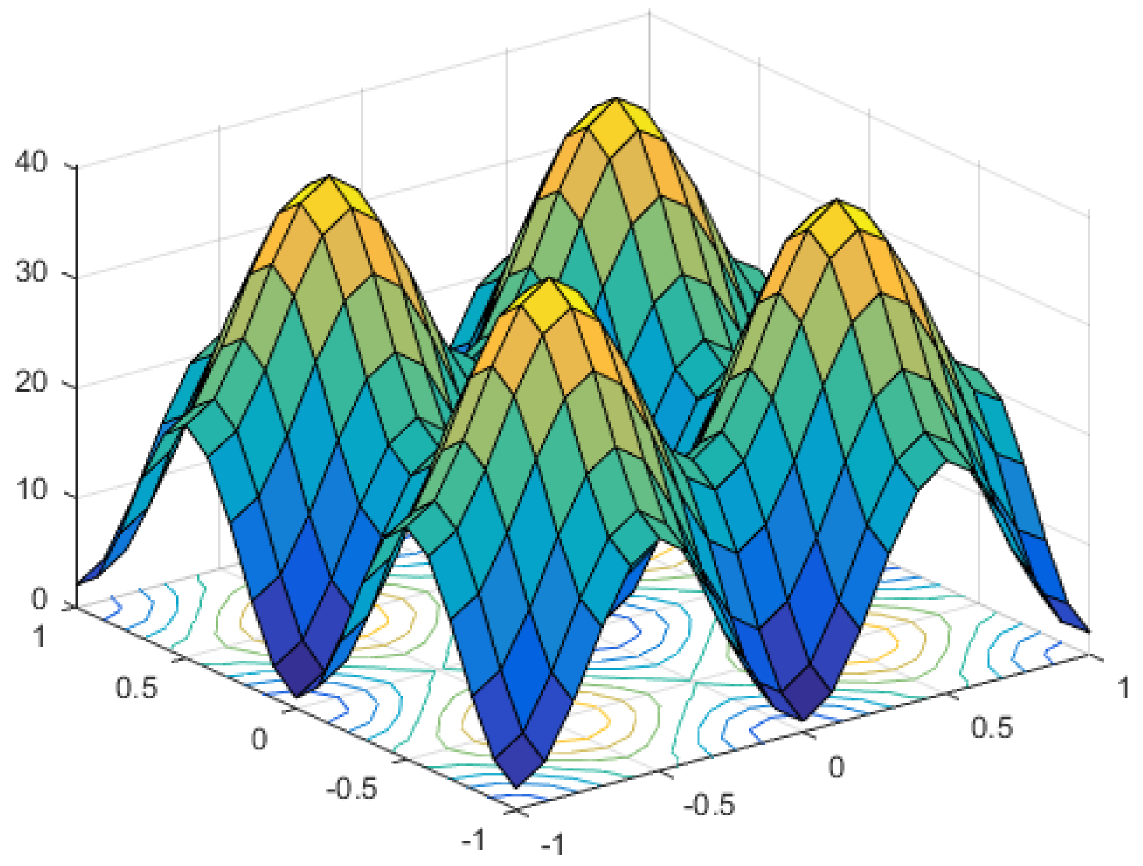


Test Problems and Comparison

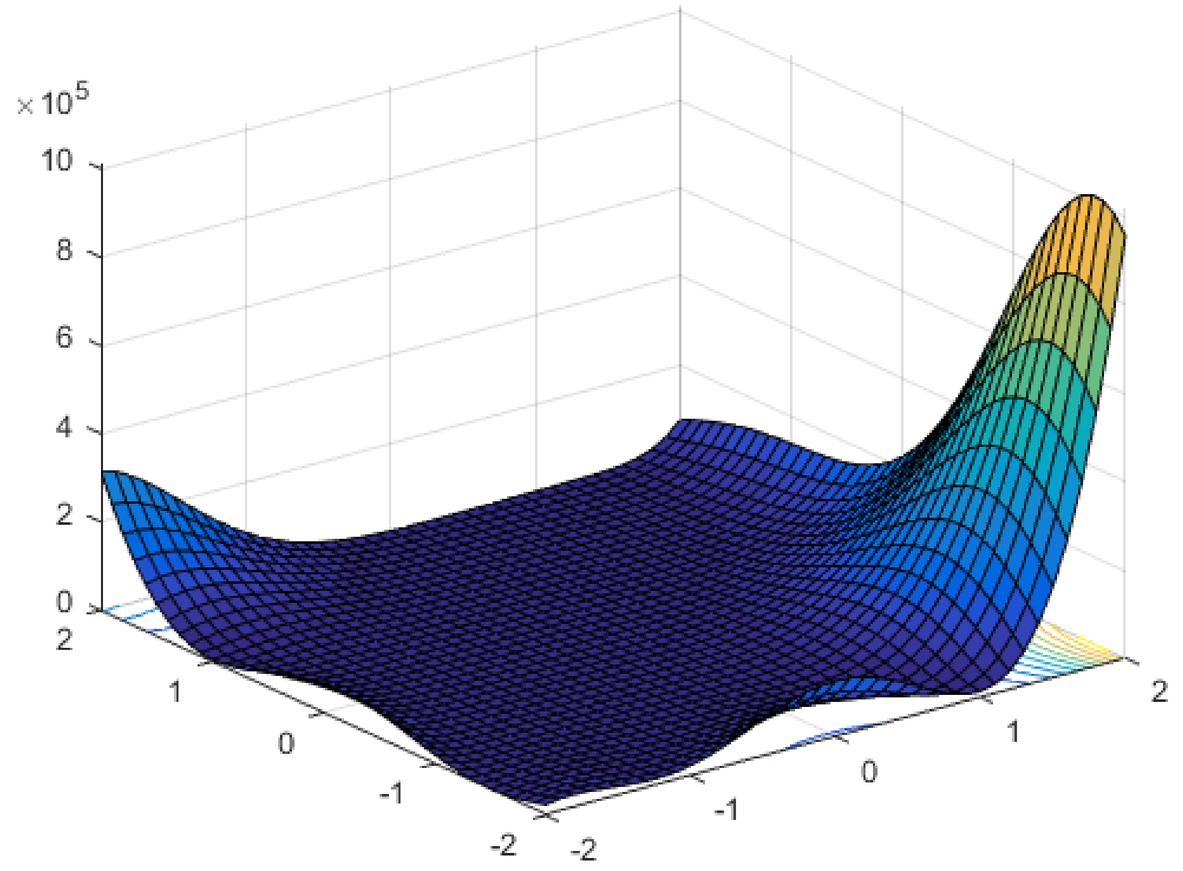
Test Functions

- 23 benchmark functions with different characteristics [2].
- Some have a relatively high dimension (e.g. 30).
- Some have many local optima.

Benchmark function f_9 at a closer look



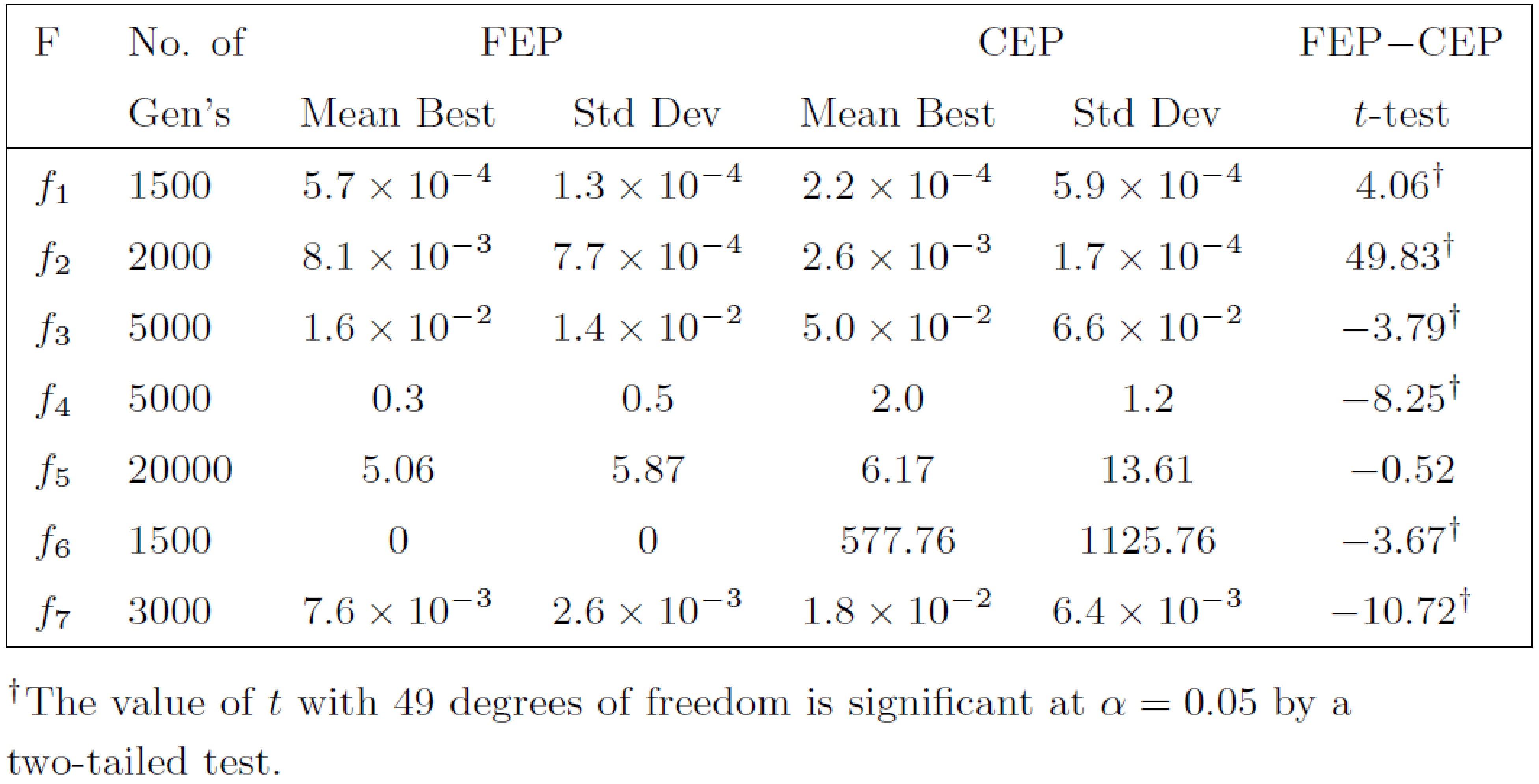
Benchmark function f_{18} at a closer look



Experimental Setup

- Population size 100.
- All experiments were run 50 times, i.e., 50 trials.
  - [Question] Why repeating many times?
- Initial populations were the same for CEP and FEP.

Results on Unimodal Functions f_1 - f_7



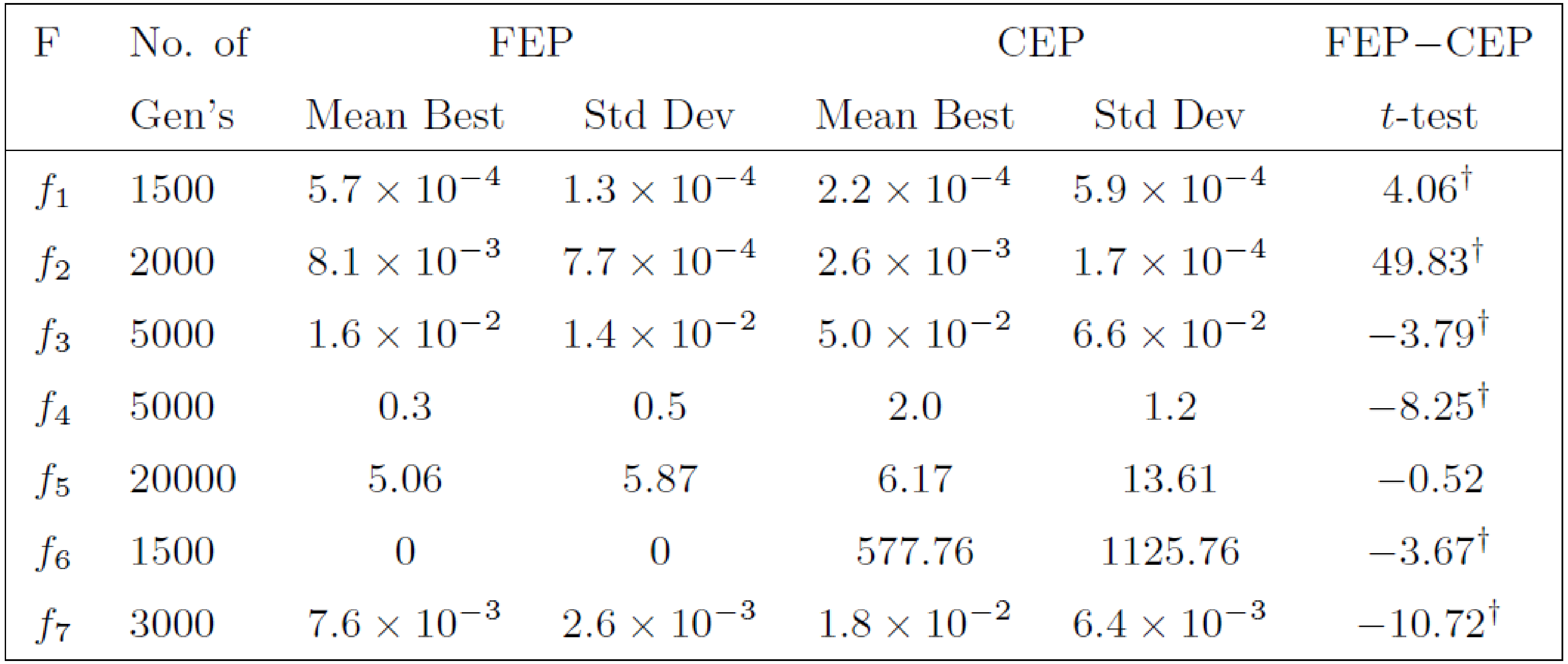
F	No. of	FEP		CEP		FEP−CEP
	Gen's	Mean Best	Std Dev	Mean Best	Std Dev	t-test
f_1	1500	5.7×10^{-4}	1.3×10^{-4}	2.2×10^{-4}	5.9×10^{-4}	$4.06^{\dagger}$
f_2	2000	8.1×10^{-3}	7.7×10^{-4}	2.6×10^{-3}	1.7×10^{-4}	$49.83^{\dagger}$
f_3	5000	1.6×10^{-2}	1.4×10^{-2}	5.0×10^{-2}	6.6×10^{-2}	$-3.79^{\dagger}$
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.

T-test

- T-test assumes that the data are normally distributed. This may not true in this case.
- A better way to do statistical tests when comparing EAs is to use a non-parametric method, e.g., Wilcoxon signed-rank test.

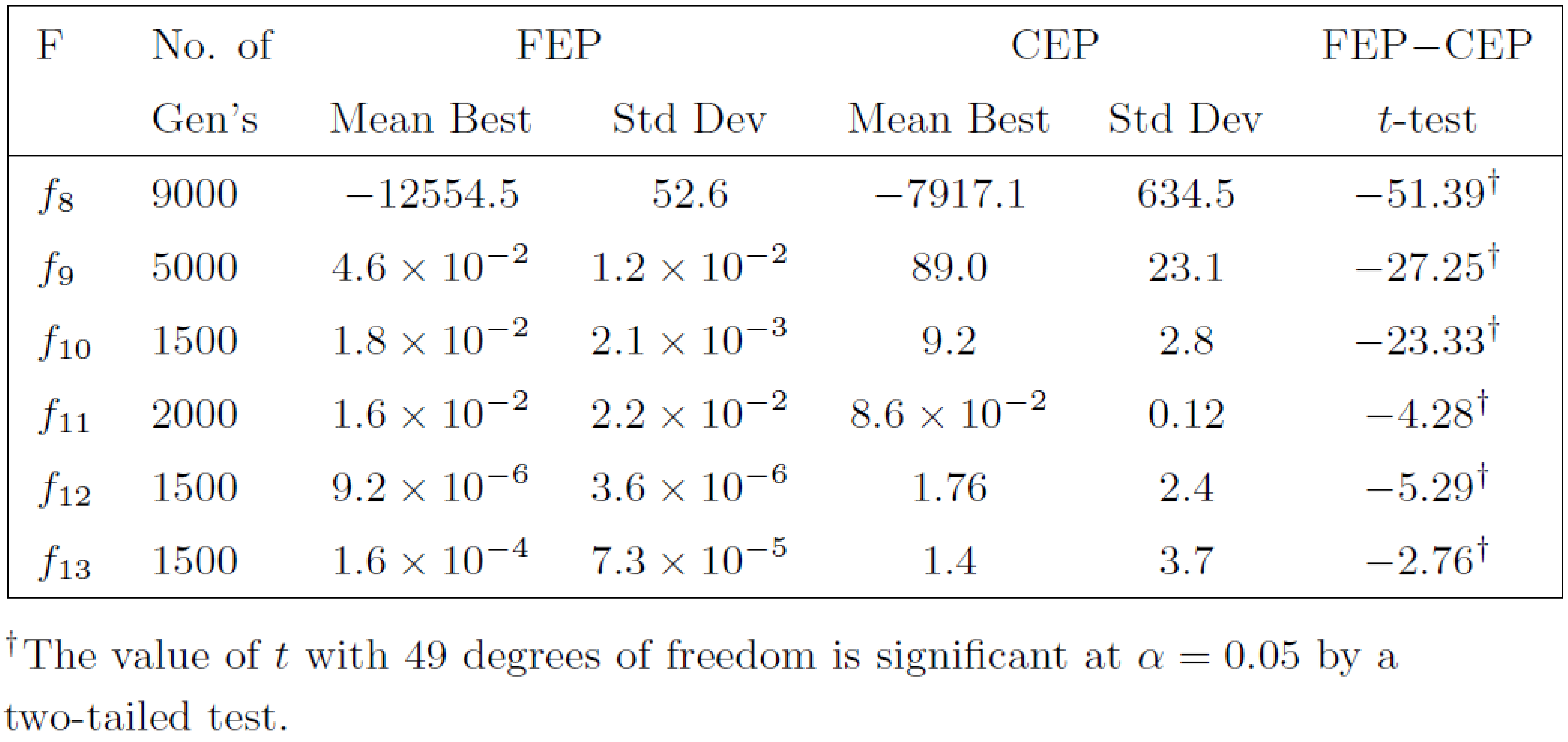
Discussions on Unimodal Functions f_1 - f_{17}



F	No. of	FEP		CEP		FEP−CEP
	Gen's	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}	$-10.72^{\dagger}$

- FEP performed better than CEP on $f_3 - f_7$.
- CEP was better for f_1 and f_2.
- FEP converged faster, even for $f_1$and f_2(for a long period).

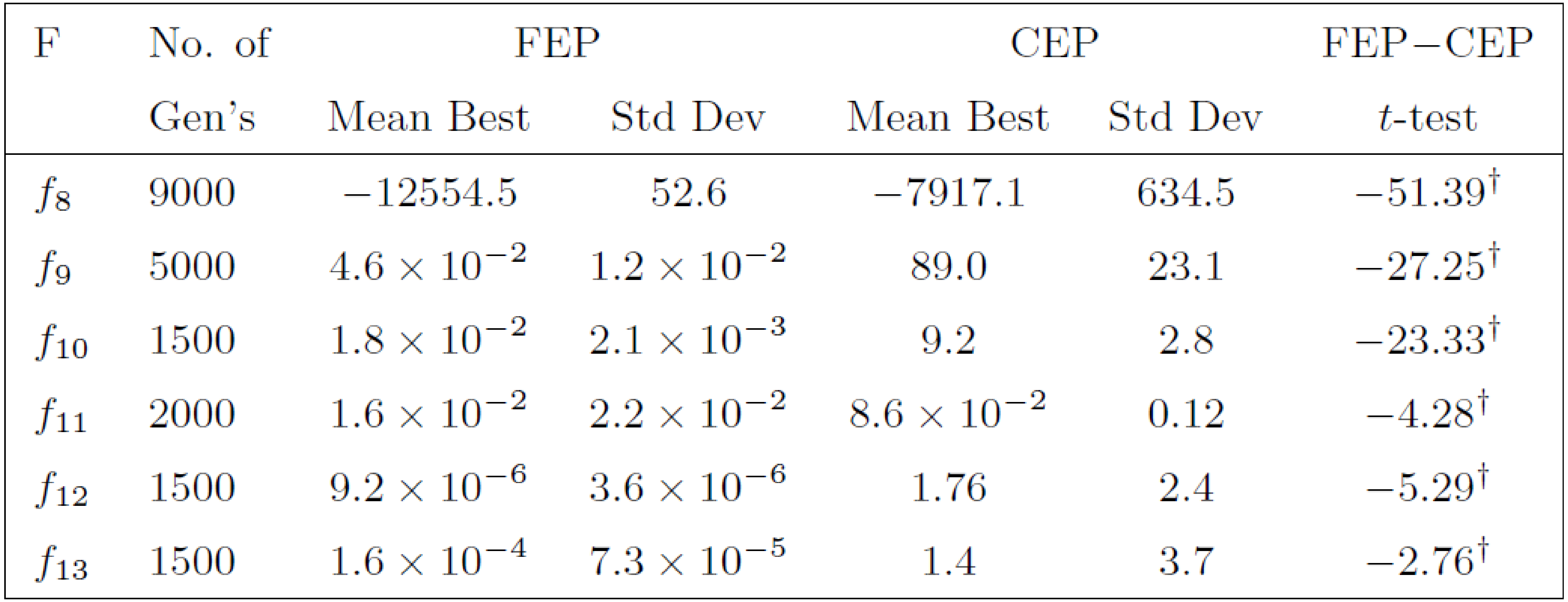
Results on Multimodal Functions f_8 - f_{13}



F	No. of	FEP		CEP		FEP−CEP
	Gen's	Mean Best	Std Dev	Mean Best	Std Dev	t-test
f_8	9000	-12554.5	52.6	-7917.1	634.5	$-51.39^{\dagger}$
f_9	5000	4.6×10^{-2}	1.2×10^{-2}	89.0	23.1	$-27.25^{\dagger}$
f_{10}	1500	1.8×10^{-2}	2.1×10^{-3}	9.2	2.8	$-23.33^{\dagger}$
f_{11}	2000	1.6×10^{-2}	2.2×10^{-2}	8.6×10^{-2}	0.12	$-4.28^{\dagger}$
f_{12}	1500	9.2×10^{-6}	3.6×10^{-6}	1.76	2.4	$-5.29^{\dagger}$
f_{13}	1500	1.6×10^{-4}	7.3×10^{-5}	1.4	3.7	$-2.76^{\dagger}$

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

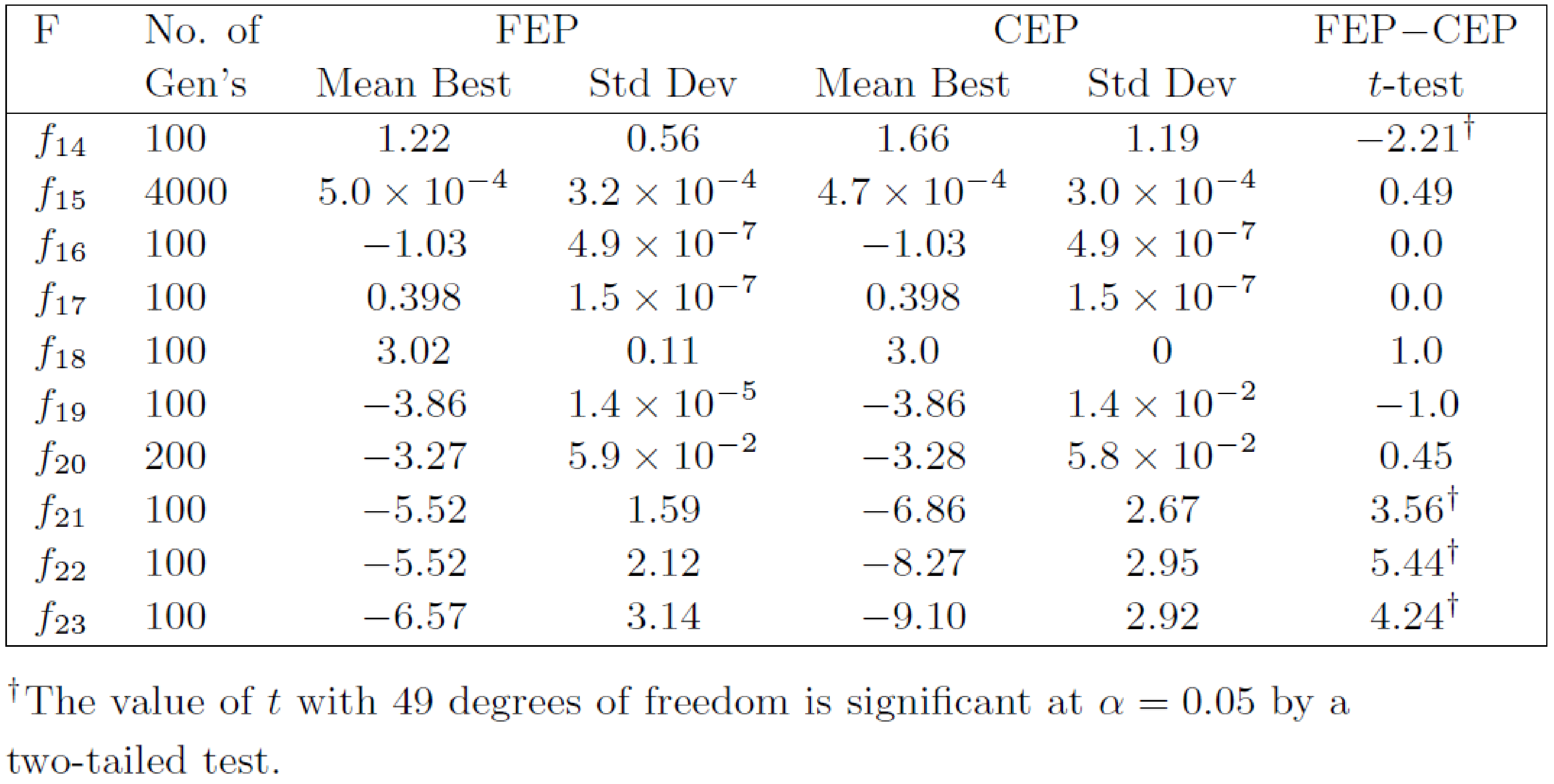
Discussions on Multimodal Functions f_8 - f_{13}



F	No. of	FEP		CEP		FEP−CEP
	Gen's	Mean Best	Std Dev	Mean Best	Std Dev	t-test
f_8	9000	-12554.5	52.6	-7917.1	634.5	$-51.39^{\dagger}$
f_9	5000	4.6×10^{-2}	1.2×10^{-2}	89.0	23.1	$-27.25^{\dagger}$
f_{10}	1500	1.8×10^{-2}	2.1×10^{-3}	9.2	2.8	$-23.33^{\dagger}$
f_{11}	2000	1.6×10^{-2}	2.2×10^{-2}	8.6×10^{-2}	0.12	$-4.28^{\dagger}$
f_{12}	1500	9.2×10^{-6}	3.6×10^{-6}	1.76	2.4	$-5.29^{\dagger}$
f_{13}	1500	1.6×10^{-4}	7.3×10^{-5}	1.4	3.7	$-2.76^{\dagger}$

- FEP converged faster to a better solution.
- FEP seemed to deal with many local minima well.

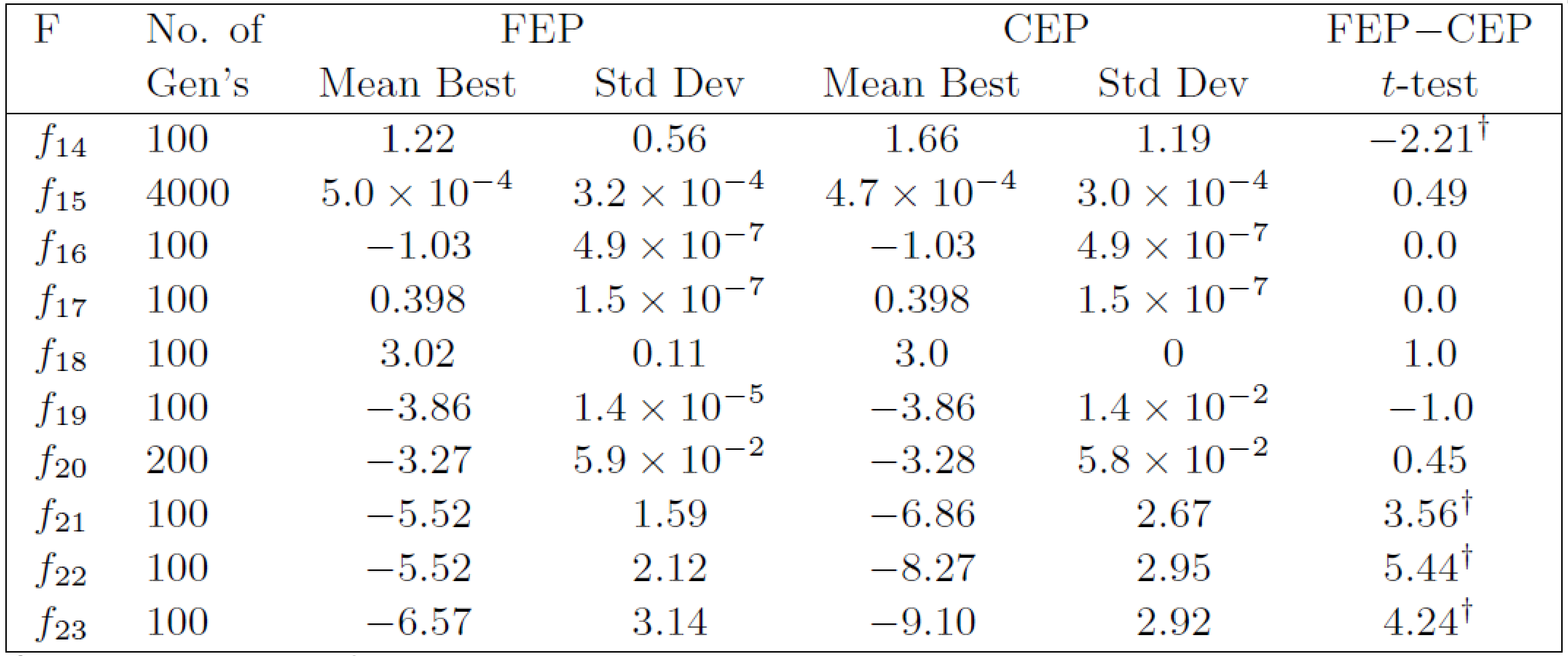
Results on Multimodal Functions f_{14} - f_{23}



F	No. of	FEP		CEP		FEP−CEP
	Gen's	Mean Best	Std Dev	Mean Best	Std Dev	t-test
f_{14}	100	1.22	0.56	1.66	1.19	$-2.21^{\dagger}$
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	$3.56^{\dagger}$
f_{22}	100	−5.52	2.12	−8.27	2.95	$5.44^{\dagger}$
f_{23}	100	−6.57	3.14	−9.10	2.92	$4.24^{\dagger}$

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

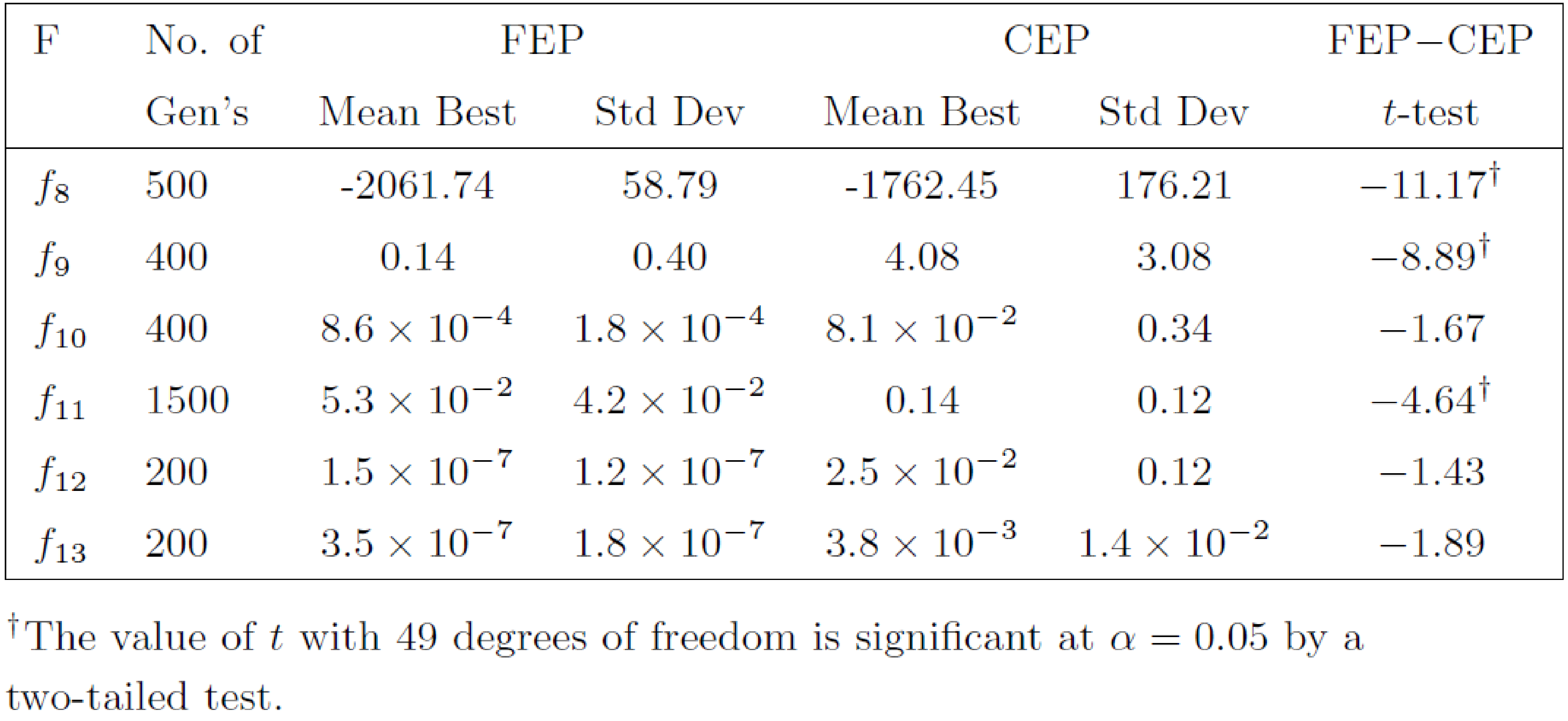
Discussion on Multimodal Functions f_{14} - f_{23}



F	No. of	FEP		CEP		FEP−CEP
	Gen's	Mean Best	Std Dev	Mean Best	Std Dev	t-test
f_{14}	100	1.22	0.56	1.66	1.19	$-2.21^{\dagger}$
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	$3.56^{\dagger}$
f_{22}	100	-5.52	2.12	-8.27	2.95	$5.44^{\dagger}$
f_{23}	100	-6.57	3.14	-9.10	2.92	$4.24^{\dagger}$

- The results are mixed!
  - FEP and CEP performed equally well on $f_{16}$and f_{17}. They are comparable on $f_{15}$and f_{18}– f_{20}.
  - CEP performed better on f_{21}– f_{23}(Shekel functions).
- Is it because the dimension was low so that CEP appeared to be better?

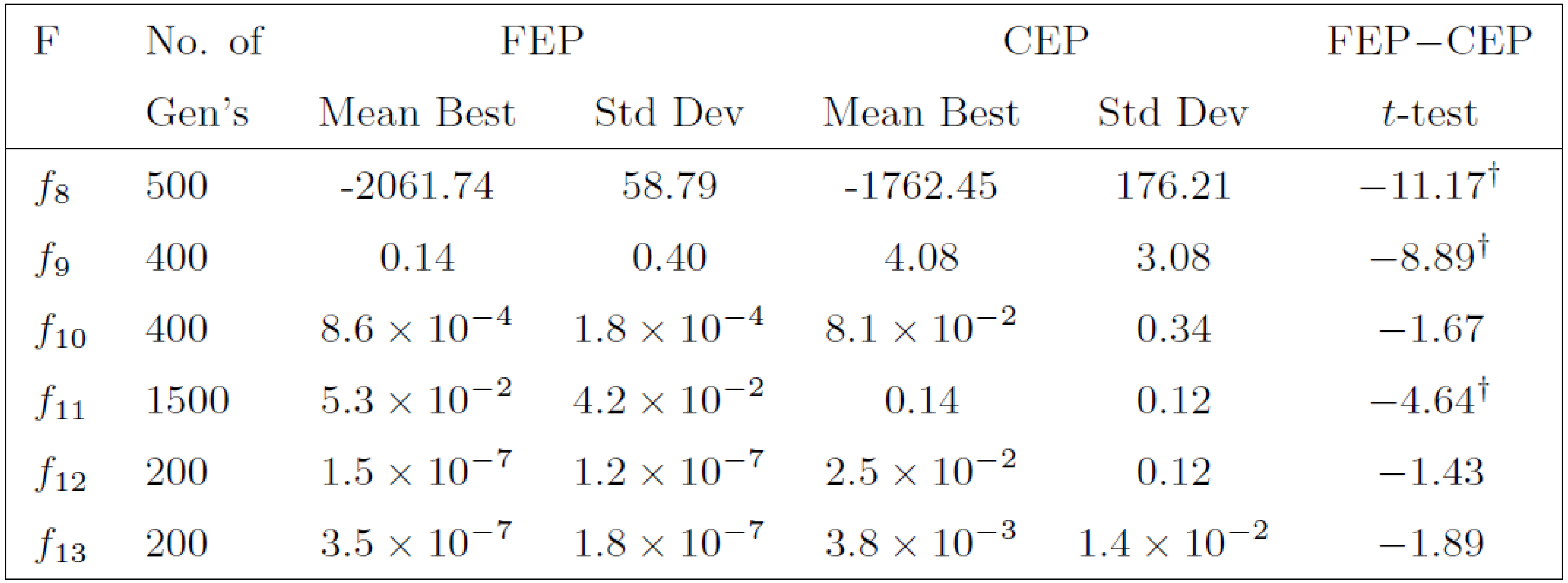
Results on Low-Dimensional Functions f_8 - f_{13}



F	No. of	FEP		CEP		FEP−CEP
	Gen's	Mean Best	Std Dev	Mean Best	Std Dev	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.

Discussion on Low-Dimensional Functions f_8 - f_{13}



F	No. of	FEP		CEP		FEP−CEP
	Gen's	Mean Best	Std Dev	Mean Best	Std Dev	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

- FEP still converged faster to better solutions.
- Dimensionality does not play a major role in causing the difference between FEP and CEP.
- There must be something inherent in those functions which caused such difference.

- How can we gain a deeper understanding of these results?

When and Why Cauchy Mutation Performed Better

Given $G(0,1)$ and $C(1)$, the expected length of Gaussian and Cauchy jumps are:

$$E_{Gaussian}(x) = \int_0^{+\infty} x \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \frac{1}{\sqrt{2\pi}} = 0.399$$

$$E_{Cauchy}(x) = \int_0^{+\infty} x \frac{1}{\pi(1+x^2)} dx = +\infty$$

It is obvious that Gaussian mutation is much localised than Cauchy mutation.

Why and When Large Jumps are Beneficial

(Only 1-d case is considered here for convenience's sake.)

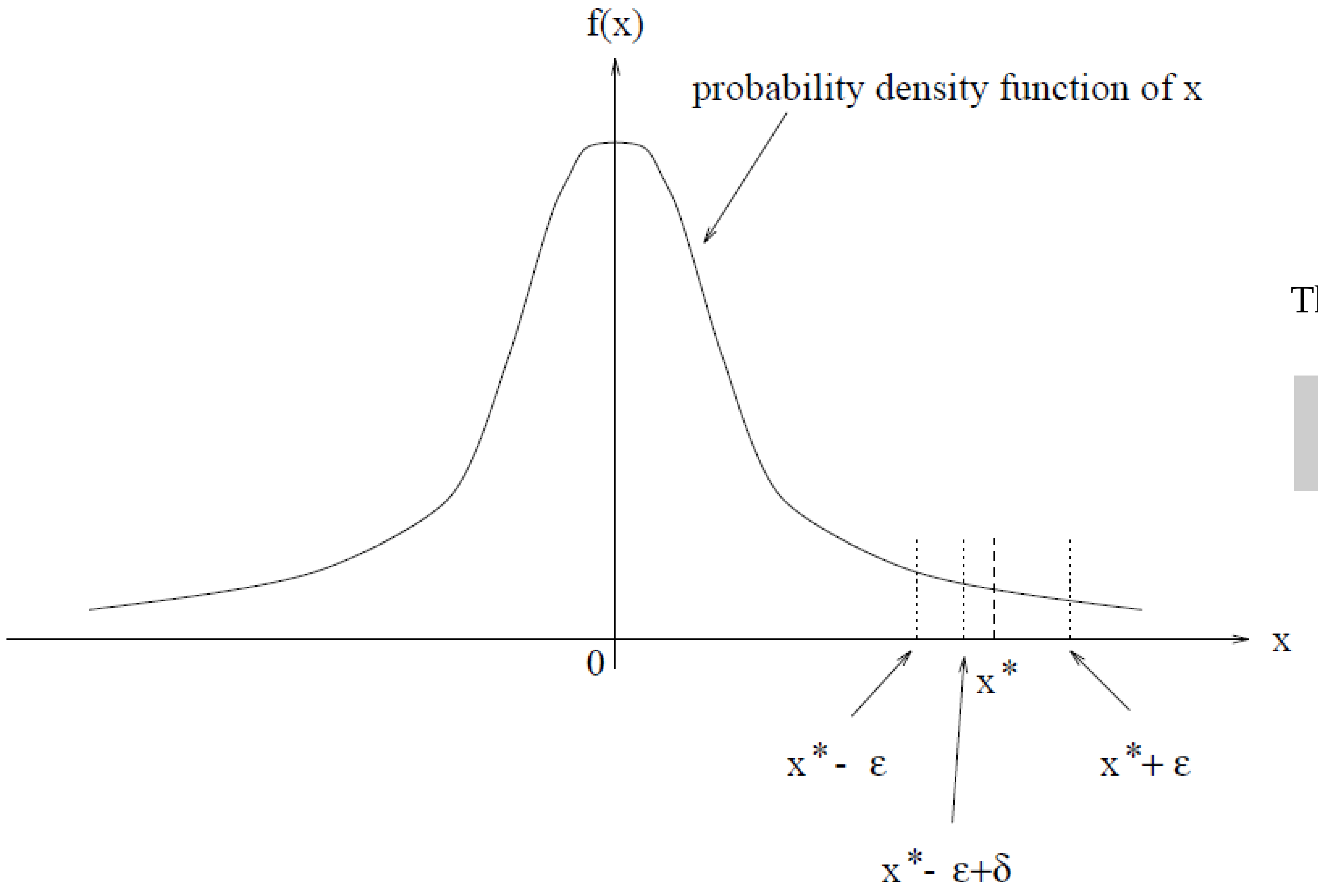
Take the Gaussian mutation with $G\left(0, \sigma^2\right)$ distribution as an example, i.e.,

$$f_{G\left(0,\sigma^2\right)}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \; -\infty < x < \infty,$$

the probability of generating a point in the neighbourhood of the global optimum $x^*$is given by

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

where $\epsilon > 0$ is the neighbourhood size and σ is often regarded as the step size of the Gaussian distribution mutation. Figure on the next page illustrates the situation.



The mean value theorem:

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

Figure 4: Evolutionary search as neighbourhood search, where x^* is the global optimum and $\epsilon > 0$ is the neighbourhood size.

An Analytical Result

It can be shown that when $|x^* - \epsilon + \delta| > \sigma$, we have

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

That is, the larger σ is, the larger $P_{G(0,\sigma^2)}(|x - x^*| \leq \epsilon)$ if $|x^* - \epsilon +$

Analysis

- 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 optimal solution and the current search point (at zero) is larger than the step size.
- As 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.

Empirical Evidence I

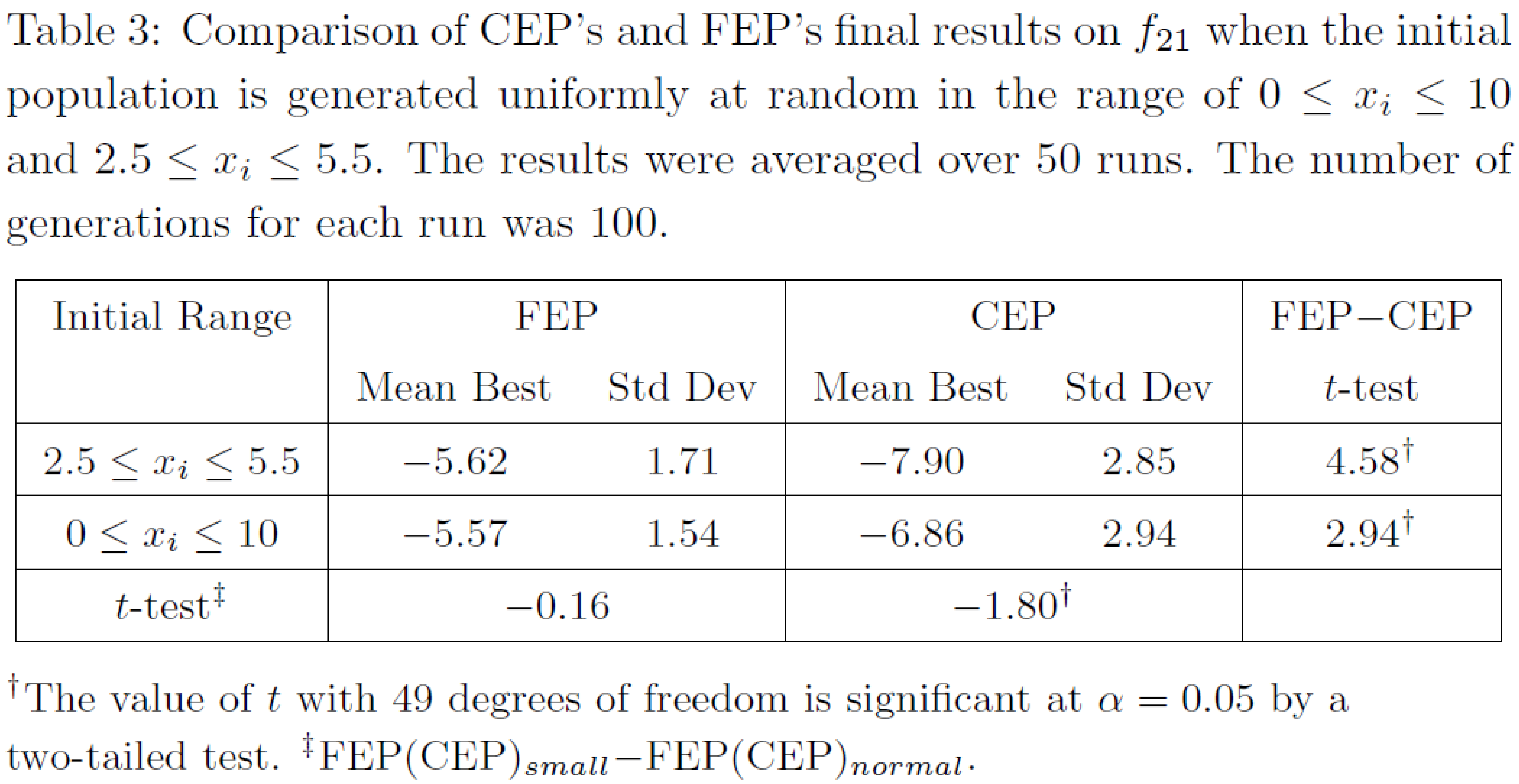


Table 3: Comparison of CEP's and FEP's final results on f_{21} when the initial population is generated uniformly at random in the range of $0 \leq x_i \leq 10$ and $2.5 \leq x_i \leq 5.5$. The results were averaged over 50 runs. The number of generations for each run was 100.

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[†]
$0 \leq x_i \leq 10$	−5.57	1.54	−6.86	2.94	2.94[†]
t-test[‡]	−0.16		−1.80[†]		

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

Empirical Evidence II

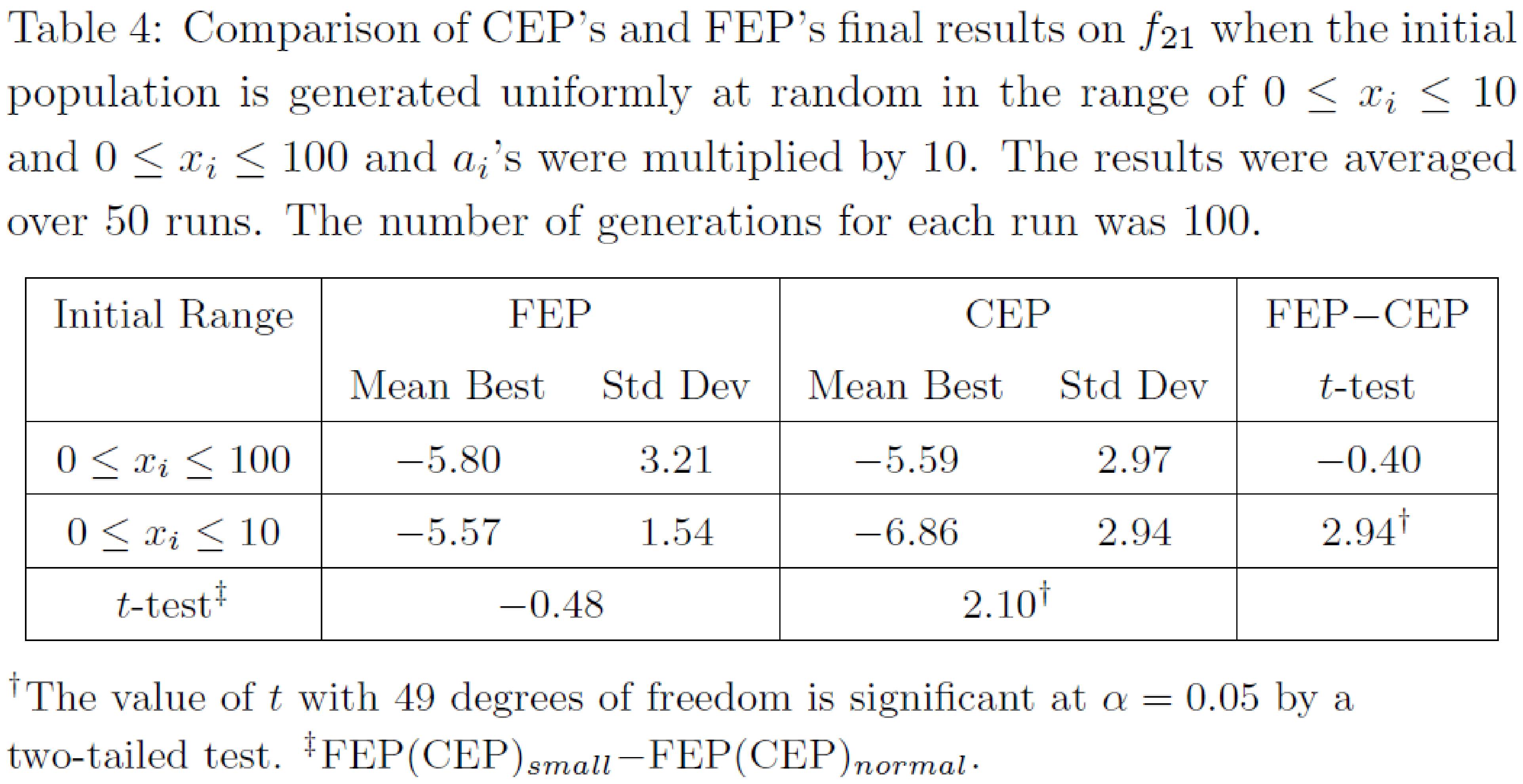


Table 4: Comparison of CEP's and FEP's final results on f_{21} when the initial population is generated uniformly at random in the range of $0 \leq x_i \leq 10$ and $0 \leq x_i \leq 100$ and a_i's were multiplied by 10. The results were averaged over 50 runs. The number of generations for each run was 100.

Initial Range	FEP		CEP		FEP−CEP
	Mean Best	Std Dev	Mean Best	Std Dev	t-test
$0 \leq x_i \leq 100$	−5.80	3.21	−5.59	2.97	−0.40
$0 \leq x_i \leq 10$	−5.57	1.54	−6.86	2.94	$2.94^{\dagger}$
t-test‡	−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. $^{\ddagger}\text{FEP(CEP)}_{small} - \text{FEP(CEP)}_{normal}$.

Summary: What Do Mutation and Self-Adaptation Do

- Cauchy mutation performs well when the global optimum is far away from the current search location. Its behaviour can be explained theoretically and empirically.
- An optimal search step size can be derived if we know where the global optimum is. Unfortunately, such information is unavailable for real-world problems.
- The performance of FEP can be improved by more suitable parameters, instead of copying CEP's parameter setting.

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

Search Step Size and Search Bias

Search Step Size and Search Bias

- The search step size of mutation is crucial in deciding the search performance.

- In general, different search operators have different search step sizes, and thus appropriate for different problems as well as different evolutionary search stages for a single problem.

- Search bias of an evolutionary search operator includes its step size and search directions. Search bias of a search operator determines how likely an offspring will be generated from parent(s).

Mixing Search Biases by Self-Adaptation

- Since the global optimum is unknown in real-world applications, it is impossible to know *a priori* what search biases we should use in EAs.
- One way to get around this problem is to use a variety of different biases and allow evolution to find out which one(s) are more promising than others.
- Rather than using either Gaussian or Cauchy mutations, we can mix them.
- That is, two candidate offspring will be generated from every parent, one by Gaussian mutation and one by Cauchy mutation. The fitter one will survive as the single child.

Other Mixing Methods

Mean mutation operator: Takes the average of the two mutations.

$$\boldsymbol{x'}_{\boldsymbol{i}}(j) = \boldsymbol{x}_{\boldsymbol{i}}(j) + \boldsymbol{\eta}_{\boldsymbol{i}}(j)(0.5(N_j(0,1) + C_j(1)))$$

where $N_j(0,1)$ is a normally distributed number while $C_j(1)$ follows a Cauchy distribution with parameter 1.

Adaptive mutation operator: It's actually a self-adaptive method.

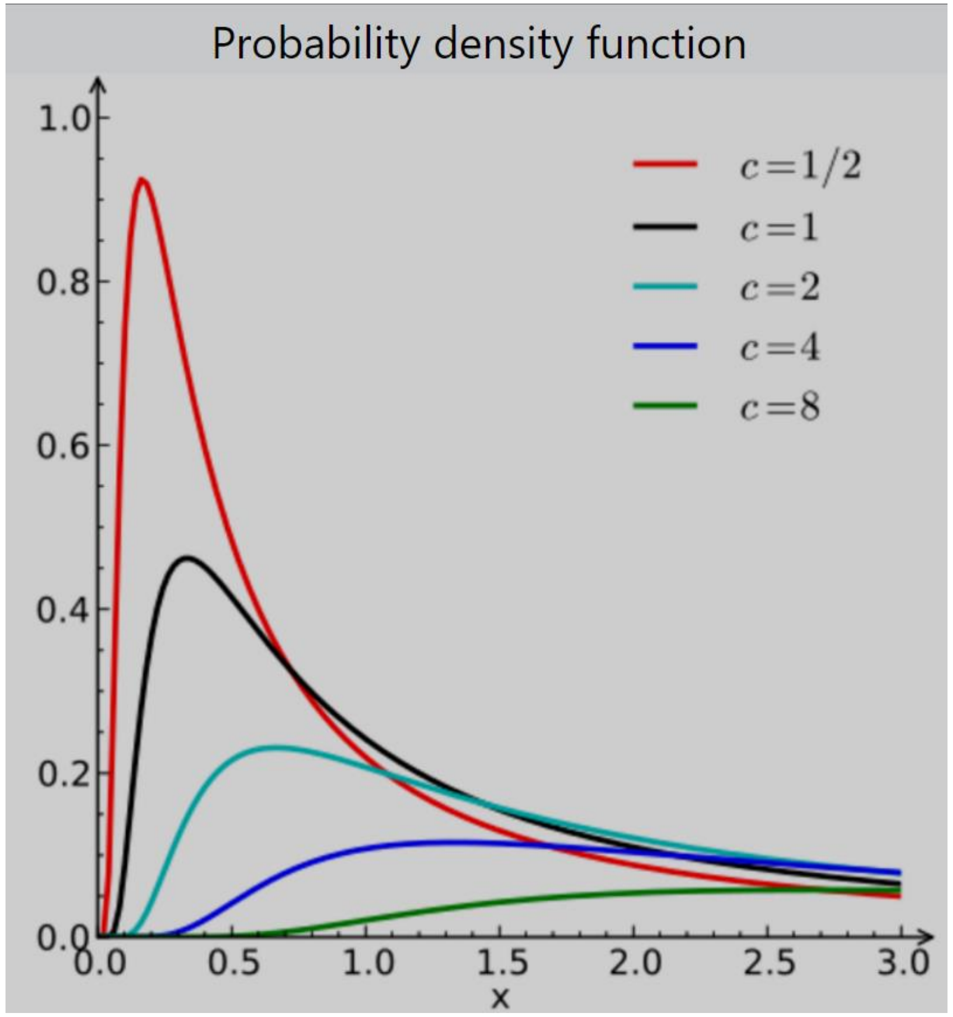
$$\boldsymbol{x'}_{\boldsymbol{i}}(j) = \boldsymbol{x}_{\boldsymbol{i}}(j) + \boldsymbol{\eta}_{\boldsymbol{1i}}(j)N_j(0,1) + \boldsymbol{\eta}_{\boldsymbol{2i}}(j)\ C_j(1)$$

where both $\boldsymbol{\eta}_{\boldsymbol{1i}}(j)$ and $\boldsymbol{\eta}_{\boldsymbol{2i}}(j)$ are self-adaptive parameters.

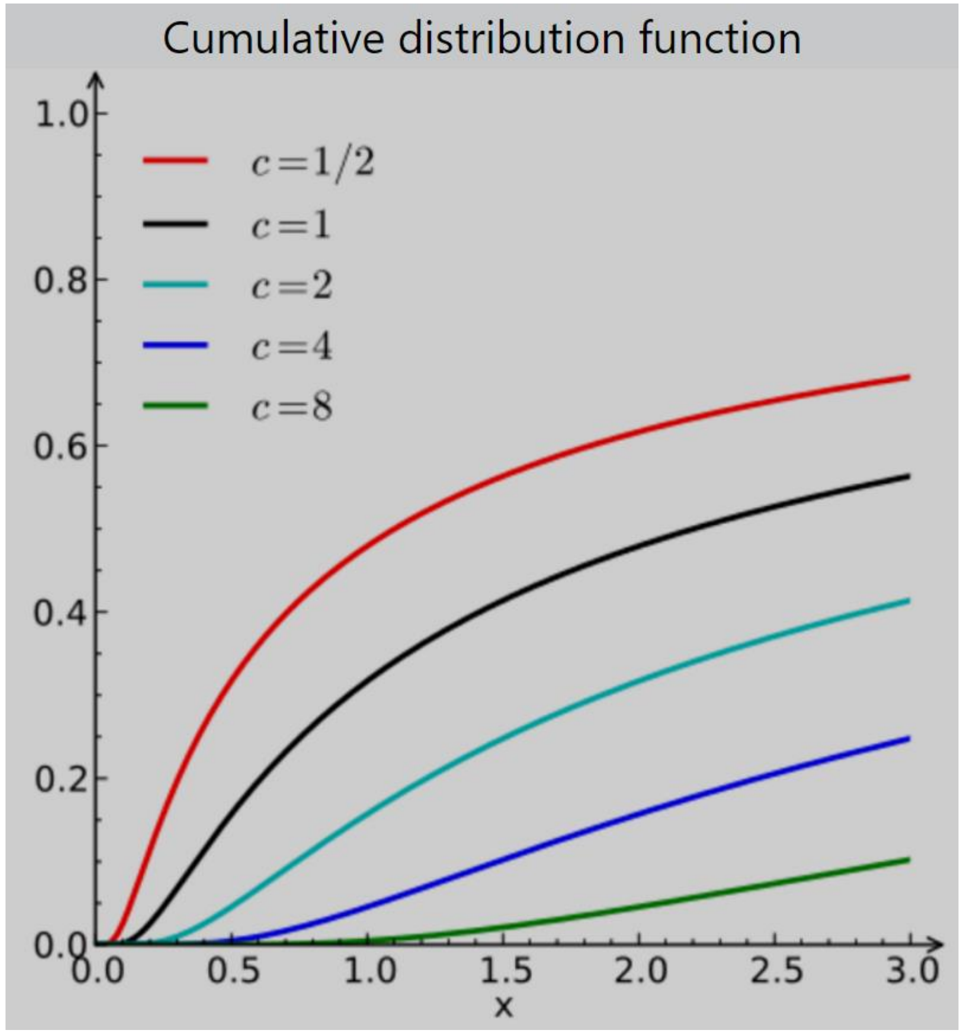
A More General Self-Adaptive Method

- The idea of mixing can be generalised to Lévy mutation.
- Lévy probability distribution can be tuned to generate any distribution between the Gaussian and Cauchy probability distributions.
- Hence we can use Lévy mutation with different parameters in EAs and let evolution to decide which one to use.

Lévy distribution



PDF	$\sqrt{\frac{c}{2\pi}}\,\frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$



Details are important

- EP or its variants are easy to implement.
- You might discover …

An Anomaly of Self-adaptation in EP

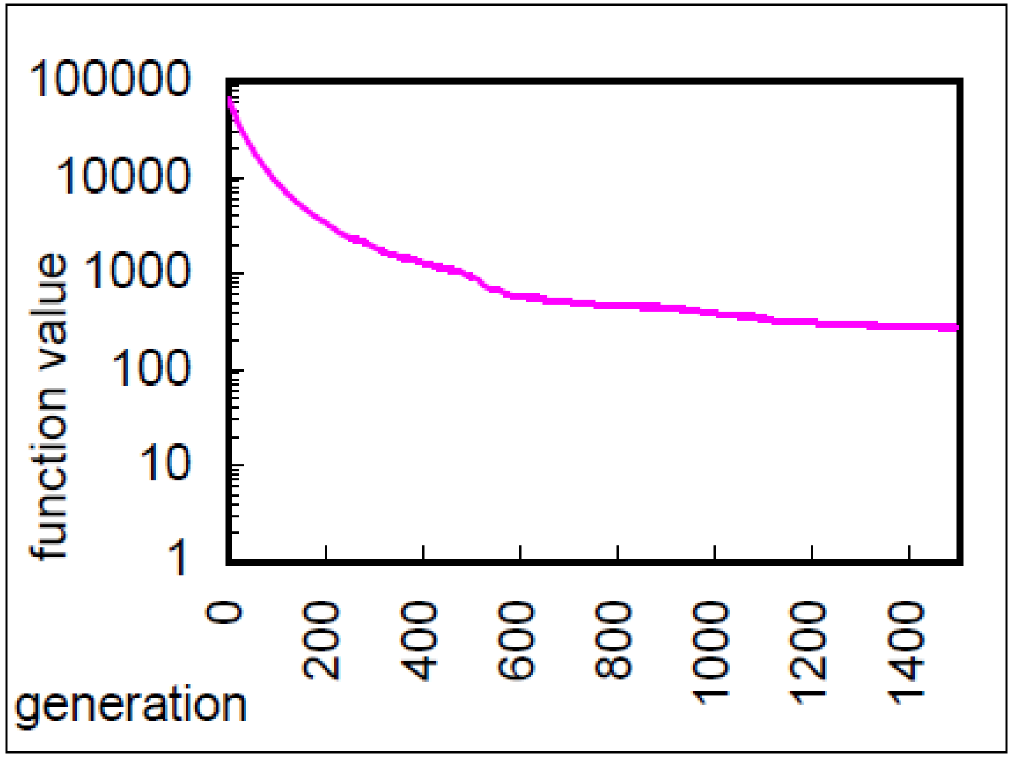


Figure 7: The 30-d sphere model stagnates early, from mean of 50 runs.

Why EP Stagnates Early

Generation	$(x_1(19), \eta_1(19))$	$f(x_1)$	$\frac{1}{\mu}\sum f(x_i)$
	$\vdots$		
300	(-14.50, 4.52E−3)	812.85	846.52
	$\vdots$		
600	(-14.50, 8.22E−6)	547.05	552.84
	$\vdots$		
1000	(-14.50, 1.33E−8)	504.58	504.59
	$\vdots$		
1500	(-14.50, 1.86E−12)	244.93	244.93

Table 5: The 19-th component and the fitness of the best individual in a run.

Getting Around the Anomaly

- Setting a lower bound!

- For example, set a fixed lower bound, e.g., 10^{-3} .

Use Mutation Step Size to Adjust Lower Bounds

Use the median of the mutation step size from all accepted (successful) offspring as the new lower bound for the next generation. Let $\delta_i(j) = \eta_i'(j)N_j(0,1)$. We first calculate the average mutation step size from all accepted (successful) offspring:

$$\bar{\delta}(j) = \frac{1}{\mu}\sum_{v=1}^{\mu} \delta_v(j), j = 1, \cdots, n,$$

where m is the number of the accepted offspring. Then, the lower bound of η for the next generation is

$$\eta_-^{t+1} = median\{\bar{\delta}(j), j = 1, \cdots, n\}.$$

Representation Is Important

Search and representation are fundamental to evolutionary search.

They go hand-in-hand.

1. Binary strings have often been used to represent individuals, e.g., integers and real numbers. However, they may not be good representations, because binary encoding of an integer or real number can introduce so-called *Hamming cliffs*.

   A Hamming cliff is formed when two numerically adjacent values have bit representations that are far apart

2. Gray coding can help, but does not solve the problem entirely. A better representation is to use integers or real numbers themselves.

   A Gray code is an encoding of numbers so that adjacent numbers have a single digit differing by 1.

Binary Code vs. Gray Code

Integer	Binary code	Gray code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Adaptive Representation: Cartesian vs Polar

- Although we have been using the Cartesian coordinates in all our examples so far, there are cases where a different representation would be more appropriate, e.g., polar coordinates.
- The idea of self-adaptation can also be used in representations, where the most suitable representation will be evolved rather than fixed in advance.
- For example, Cartesian and polar representations can be mixed adaptively in an EAs so that evolution can select which representation is the best in the current stage of evolutionary search.

Summary

- Search step size is a crucial factor in determining EA’s performance.
- Different operators, and EAs in general, have different search biases.
- Mixing different operators and representations adaptively can lead to better performance for many (but not all) problems.
- However, cares must be taken as self-adaptation does not always work as claimed.

Reading Materials

Books

[b1] Eiben, A. E., & Smith, J. E. (2003). *Introduction to evolutionary computing* (Vol. 53). Berlin: springer. (pages 1-48)

Articles

[1] K. Chellapilla, *Combining mutation operators in evolutionary programming*, IEEE Transactions on Evolutionary Computation, 2(3):91-96, Sept. 1998.

[2] Yao, X., Liu, Y., & Lin, G. (1999). *Evolutionary programming made faster.* IEEE Transactions on Evolutionary computation, 3(2), 82-102. K. H.

[3] T. Schnier and X. Yao, *Using Multiple Representations in Evolutionary Algorithms*, Proceedings of the 2000 Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, USA, July 2000. pp.479-486.

[4] Liang, X. Yao and C. S. Newton, *Adapting self-adaptive parameters in evolutionary algorithms*, Applied Intelligence, 15(3):171-180, November/December 2001.

[5] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by simulated annealing*. science, 220(4598), 671-680.

[6] Holland, J. H. (1992). *Genetic algorithms*. Scientific American, 267(1), 66-73.

[7] Fogel, D. B., Fogel, L. J., & Atmar, J. W. (1991, November). *Meta-evolutionary programming*. In Signals, systems and computers, 1991. 1991 Conference record of the twenty-fifth Asilomar Conference on (pp. 540-545). IEEE.