Self-Adaptive Simulated Binary Crossover for Real-Parameter Optimization

Kalyanmoy Deb*, S Karthik*, Tatsuya Okabe*

GECCO 2007

*Indian Inst. India

**Honda Research Inst., Japan

Reviewed by

Paskorn Champrasert
paskorn@cs.umb.edu
http://dssg.cs.umb.edu

Outline

- Simulated Binary Crossover (SBX Crossover)
  (revisit)
- Problem in SBX
- Self-Adaptive SBX
- Simulation Results
  - Self-Adaptive SBX for Single-OOP
  - Self-Adaptive SBX for Multi-OOP
- Conclusion

SBX
Simulated Binary Crossover

- Simulated Binary Crossover (SBX) was proposed in 1995 by Deb and Agrawal.

- SBX was designed with respect to the one-point crossover properties in binary-coded GA.
  - **Average Property:**
    The average of the decoded parameter values is the same before and after the crossover operation.
  - **Spread Factor Property:**
    The probability of occurrence of spread factor $\beta \approx 1$ is more likely than any other β value.

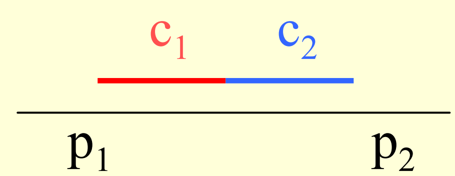
Important Property of One-Point Crossover

Spread factor:

Spread factor β is defined as the ratio of the spread of offspring points to that of the parent points:

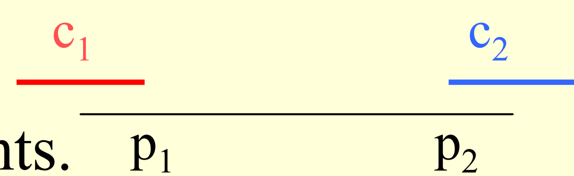
$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right|$$

- **Contracting Crossover** $\beta < 1$



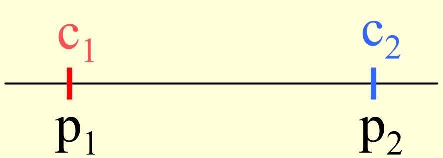
The offspring points are enclosed by the parent points.

- **Expanding Crossover** $\beta > 1$



The offspring points enclose by the parent points.

- **Stationary Crossover** $\beta = 1$



The offspring points are the same as parent points

Spread Factor (β) in SBX

- The probability distribution of β in SBX should be similar (e.g. same shape) to the probability distribution of β in Binary-coded crossover.

- The probability distribution function is defined as:

$$
\begin{cases}
c(\beta) = 0.5(n_c + 1)\beta^{n_c}, \beta \le 1 \\
c(\beta) = 0.5(n_c + 1)\frac{1}{\beta^{n_c+2}}, \beta > 1
\end{cases}
$$

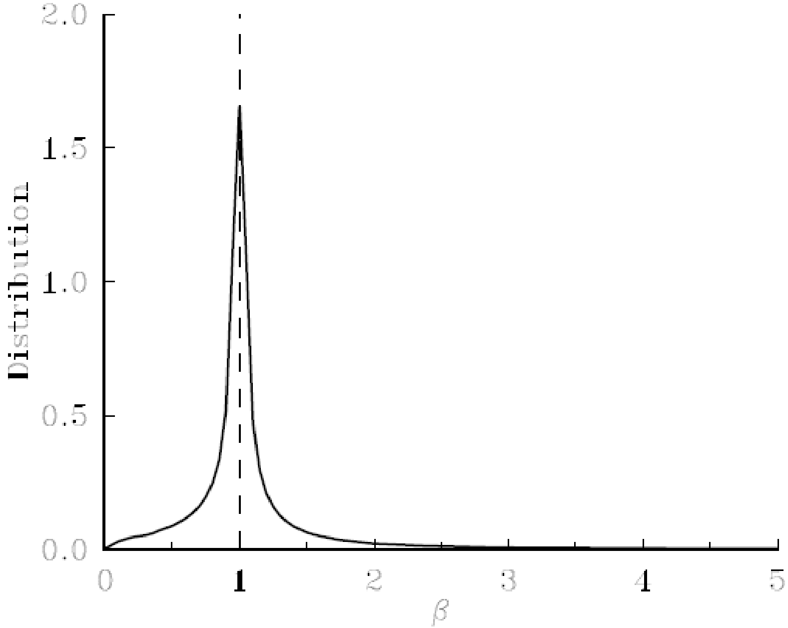
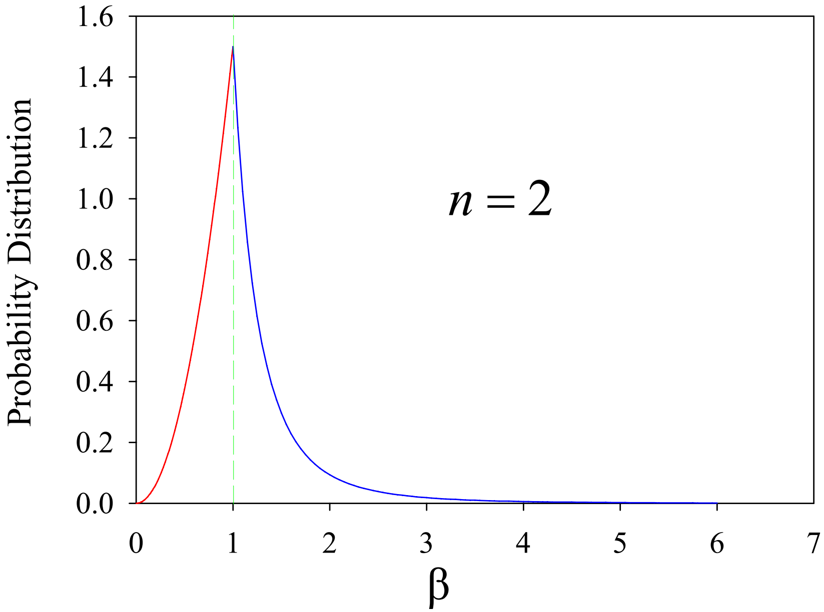


Figure 4: Probability distributions of contracting and expanding crossovers on all pairs of random binary strings of length 15 are shown.



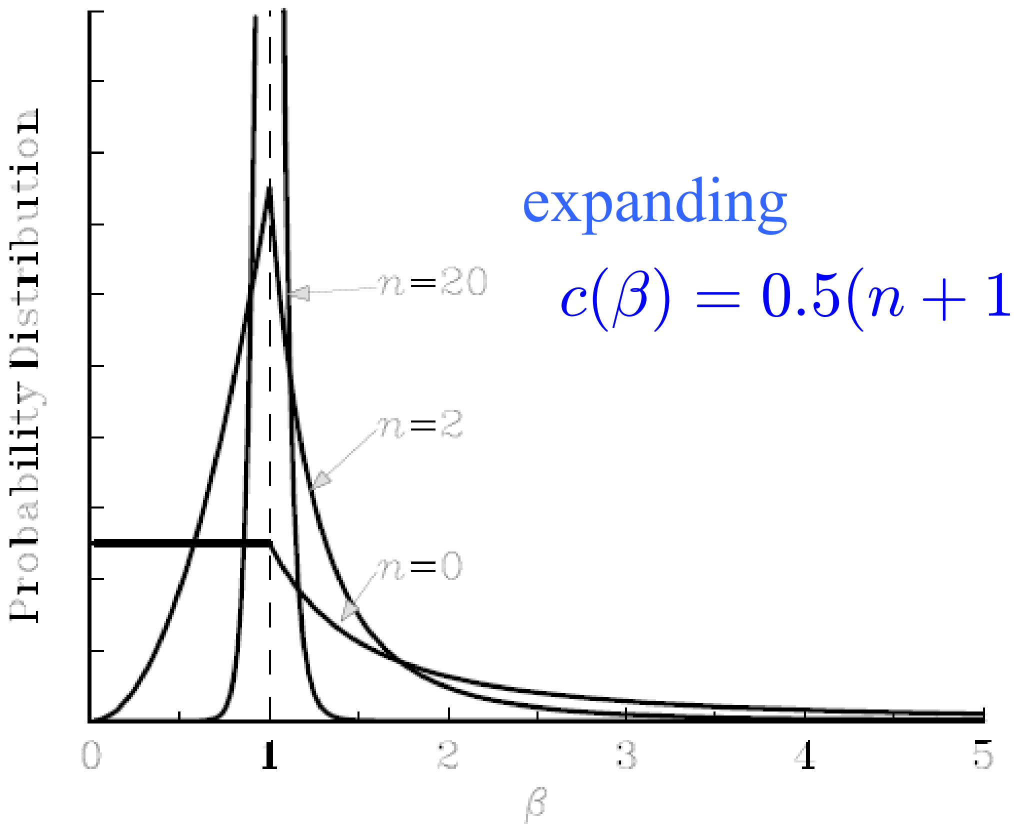
A large value of n gives a higher probability for creating 'near-parent' solutions.

contracting

$$c(\beta) = 0.5(n+1)\beta^n, \beta \le 1$$

expanding

$$c(\beta) = 0.5(n+1)\frac{1}{\beta^{n+2}}, \beta > 1$$

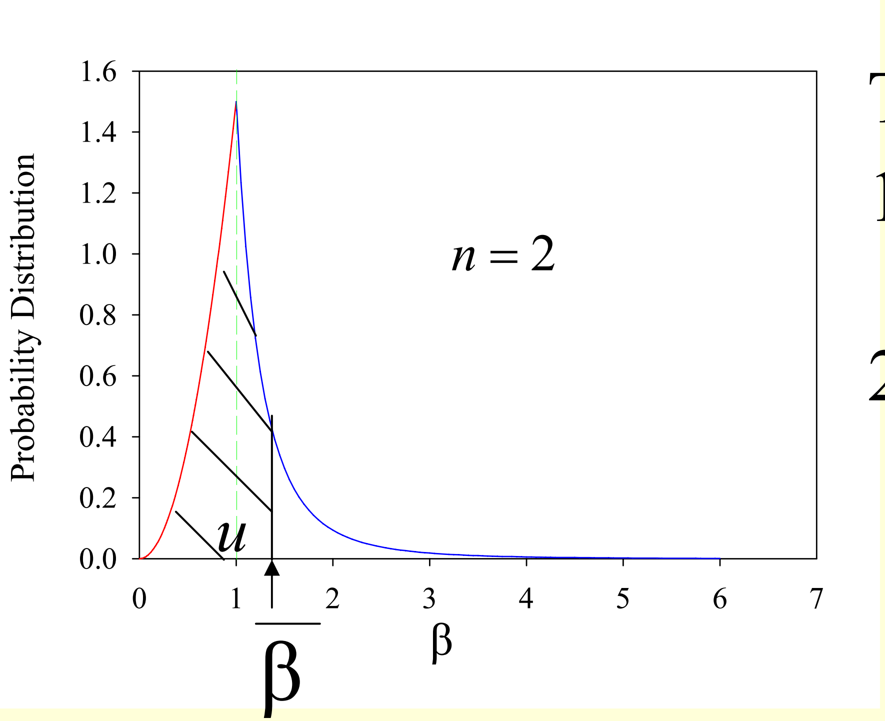


Mostly, n=2 to 5

Figure 5: Probability distributions of contracting and expanding crossovers for the proposed polynomial model are shown.

Spread Factor (β) as a random number

- $\overline{\beta}$ is a random number that follows the proposed probability distribution function:



To get a $\overline{\beta}$ value,

1) A unified random number u is generated [0,1]
2) Get β value that makes the area under the curve = u

Offspring:

$$c_1 = \bar{x} - \frac{1}{2}\overline{\beta}(p_2 - p_1)$$

$$c_2 = \bar{x} + \frac{1}{2}\overline{\beta}(p_2 - p_1)$$

SBX Summary

- Simulated Binary Crossover (SBX) is a real-parameter combination operator which is commonly used in evolutionary algorithms (EA) to optimization problems.
  - SBX operator uses two parents and creates two offspring.
  - SBX operator uses a parameter called the distribution index (n_c) which is kept fixed (i.e., 2 or 5) throughout a simulation run.
    - If n_c is large the offspring solutions are close to the parent solutions
    - If n_c is small the offspring solutions are away from the parent solutions
  - The distribution index (n_c) has a direct effect in controlling the spread of offspring solutions.

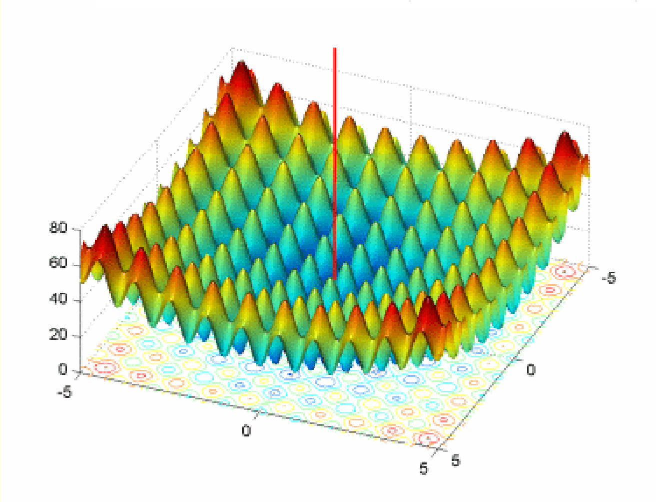
Research Issues

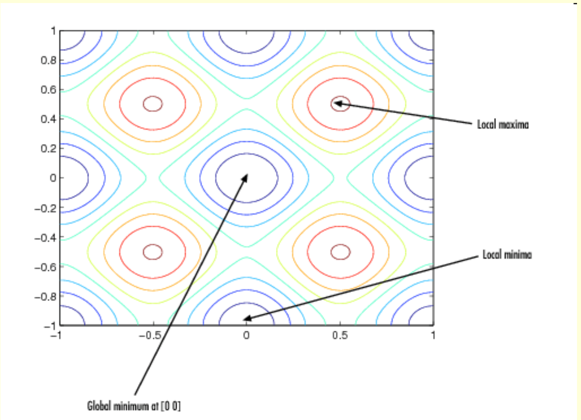
- Finding the distribution index (n_c) to an appropriate value is an important task.
  - The distribution index (n_c) has effect in
    1) Convergence speed

       if n_c is very high the offspring solutions are very close to the parent solutions the convergence speed is very low.
    2) Local/Global optimum solutions.

       Past studies of SBX crossover GA found that it cannot work well in complicated problems (e.g., Restrigin's function)

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2).$$



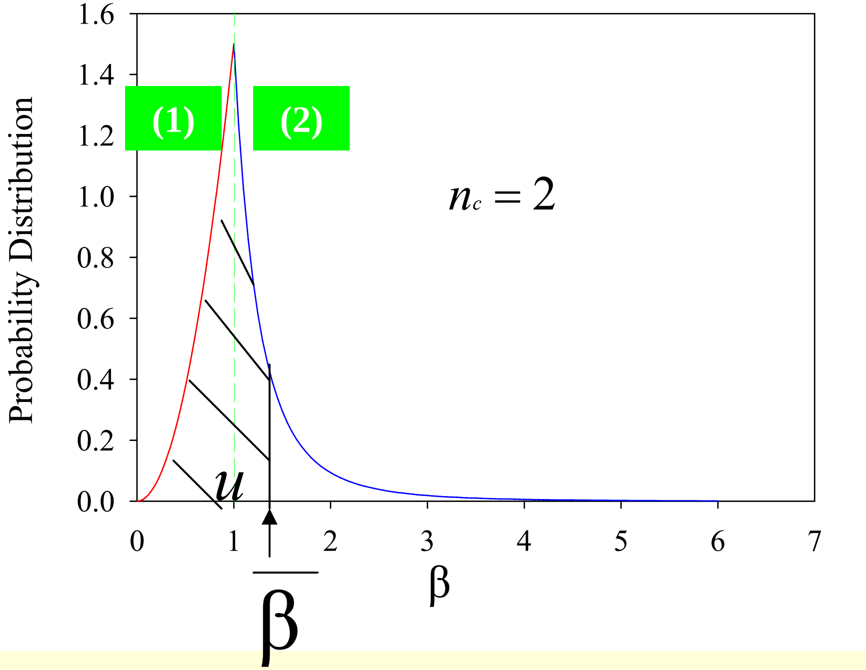


Self-Adaptive SBX

- This paper proposed a procedure for updating the distribution index (n_c) adaptively to solve optimization problems.
  - The distribution index (n_c) is updated every generation.
  - How the distribution index of the next generation (n_c) is updated (increased or decreased) depends on how the offspring outperform (has better fitness value) than its two parents.
- The next slides will show the process to make an equation to update the distribution index.

Probability density per offspring

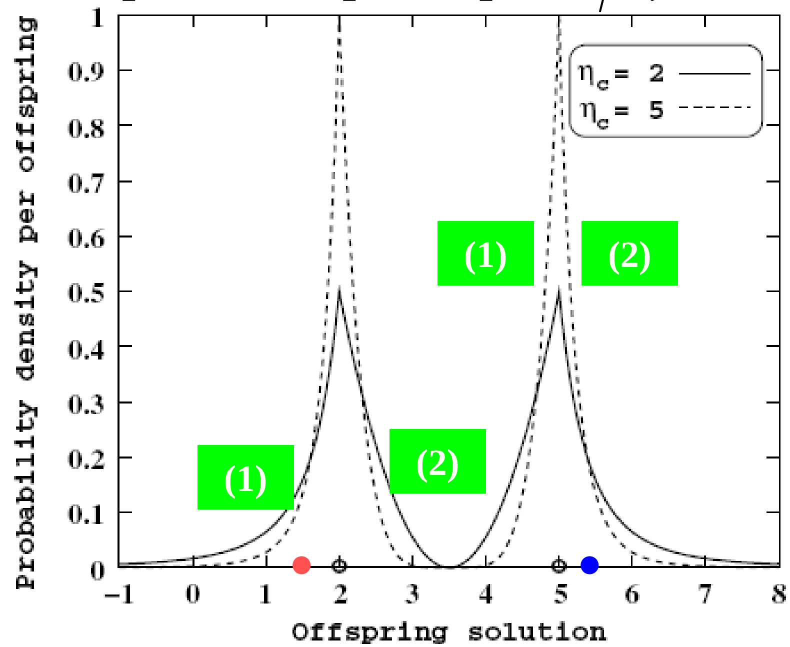
Spread factor distribution



(1) $c(\beta) = 0.5(n_c + 1)\beta^{n_c}, \beta \le 1$

(2) $c(\beta) = 0.5(n_c + 1)\frac{1}{\beta^{n_c+2}}, \beta > 1$

An example, when p1=2, p2=5 $\bar{\beta} > 1$



Offspring:

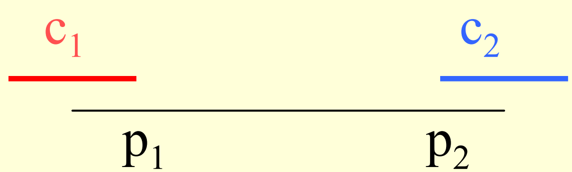
$$c_1 = \bar{x} - \frac{1}{2}\bar{\beta}(p_2 - p_1)$$

$$c_2 = \bar{x} + \frac{1}{2}\bar{\beta}(p_2 - p_1)$$

Self-Adaptive SBX (1)

Consider the case that β > 1:

$$\beta > 1$$



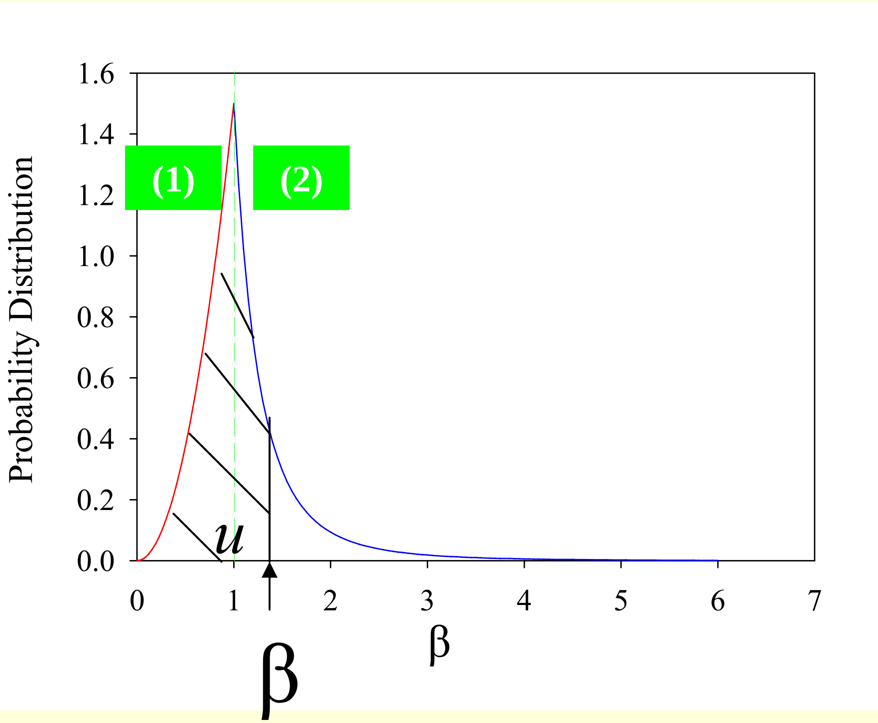
$$\beta = \left| \frac{c_2 - c_1}{p_2 - p_1} \right|$$

$$\beta = \frac{(c_2 - p_2) + (p_2 - p_1) + (p_1 - c_1)}{p_2 - p_1}$$

$$(c_2 - p_2) = (p_1 - c_1)$$

$$\boxed{\beta = 1 + \frac{2(c_2 - p_2)}{p_2 - p_1} \quad \text{--- (1)}}$$

Self-Adaptive SBX (2)



(1) $c(\beta) = 0.5(n_c + 1)\beta^{n_c}, \beta \le 1$

(2) $c(\beta) = 0.5(n_c + 1)\frac{1}{\beta^{n_c+2}}, \beta > 1$

(2)

$$\int_1^{\beta} \frac{0.5(n_c+1)}{\beta^{n_c+2}} d\beta = (u - 0.5) \qquad \text{--- (2)}$$

Self-Adaptive SBX (3)

(2)

$$\int_1^\beta \frac{0.5(n_c+1)}{\beta^{n_c+2}} d\beta = (u - 0.5) \quad \text{--- (2)}$$

$$\int_1^\beta 0.5(n_c+1)\beta^{-(n_c+2)} d\beta = (u - 0.5)$$

$$-0.5\beta^{-n_c-1} - (-0.5) = (u - 0.5)$$

$$-0.5\beta^{-n_c-1} = (u - 1)$$

$$\log(\beta^{-n_c-1}) = \log -\frac{(u-1)}{0.5}$$

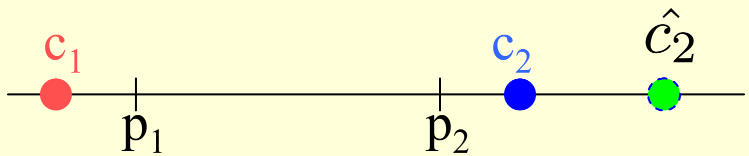
$$(-n_c - 1)\log\beta = \log(-2(u-1))$$

$$(-n_c - 1) = \frac{\log 2(1-u)}{\log\beta}$$

$$n_c = -(1 + \frac{\log 2(1-u)}{\log\beta}) \quad \text{--- (3)}$$

Self-Adaptive SBX (4)

$\beta > 1$

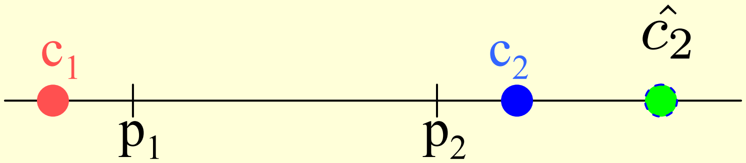


- If c_2 is better than both parent solutions,
  - the authors assume that the region in which the child solutions is created is better than the region in which the parents solutions are.
  - the authors intend to extend the child solution further away from the closet parent solution (p_2) by a factor α ($\alpha > 1$)

$$(\hat{c}_2 - p_2) = \alpha(c_2 - p_2) \quad \text{--- (4)}$$

β Update

$$\beta > 1$$



$$\beta = 1 + \frac{2(c_2 - p_2)}{p_2 - p_1} \quad \text{--- (1)}$$

$$(\hat{c}_2 - p_2) = \alpha(c_2 - p_2) \quad \text{--- (4)}$$

from (1) $$\hat{\beta} = 1 + \frac{2(\hat{c}_2 - p_2)}{p_2 - p_1}$$

$$= 1 + \frac{2((\alpha(c_2 - p_2) + p_2) - p_2)}{p_2 - p_1}$$

$$= 1 + \boxed{\frac{\alpha 2(c_2 - p_2)}{p_2 - p_1}}$$

$$\beta - 1$$

$$\hat{\beta} = 1 + \alpha(\beta - 1) \quad \text{--- (5)}$$

Distribution Index (n_c) Update

$$n_c = -(1 + \frac{\log 2(1-u)}{\log \beta}) \quad \text{--- (3)}$$

$$\hat{\beta} = 1 + \alpha(\beta - 1) \quad \text{--- (5)}$$

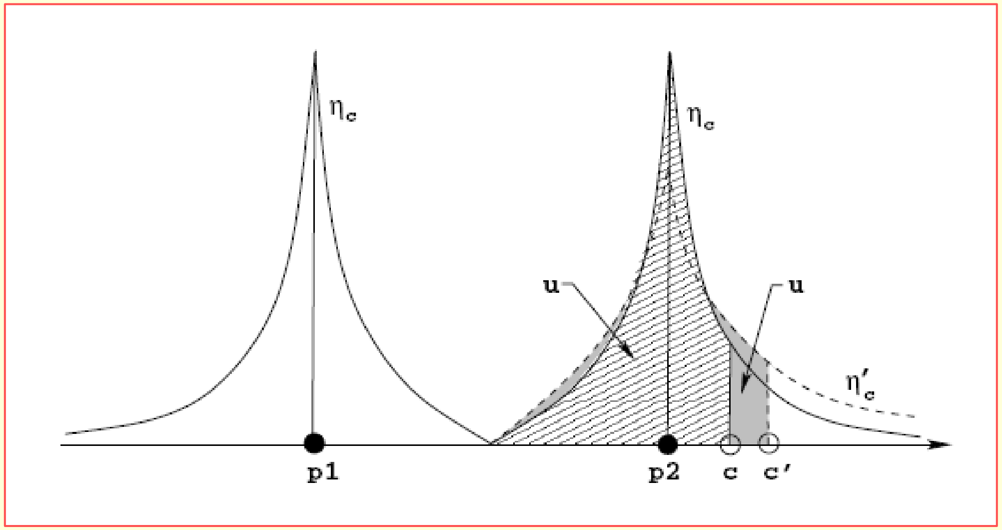
$$\hat{n_c} = -(1 + \frac{\log 2(1-u)}{\log \hat{\beta}})$$

$$\hat{n_c} = -(1 + \frac{\log 2(1-u)}{\log(1+\alpha(\beta-1))})$$

$$\log 2(1-u) = -(n_c + 1)\log \beta$$

$$\hat{n_c} = -1 + \frac{(n_c+1)\log \beta}{\log(1+\alpha(\beta-1))} \quad \text{--- (6)}$$

Distribution Index (n_c) Update



When c **is better than** p_1 and p_2 then the distribution index for the next generation is calculated as (6). The distribution index will be decreased in order to create an offspring (c') further away from its parent(p_2) than the offspring (c) in the previous generation.

$$\hat{n_c} = -1 + \frac{(n_c+1)\log\beta}{\log(1+\alpha(\beta-1))} \quad \text{--- (6)}$$

[0,50]

α is a constant (α >1)

Distribution Index (n_c) Update

When c **is worse than** p_1 and p_2 then the distribution index for the next generation is calculated as (7). The distribution index will be increased in order to create an offspring (c') closer to its parent(p_2) than the offspring (c) in the previous generation.

The 1/α is used instead of α

$$\hat{n}_c = -1 + \frac{(n_c+1)\log\beta}{\log(1+\alpha(\beta-1))} \quad \text{--- (6)}$$

$$\hat{n}_c = -1 + \frac{(n_c+1)\log\beta}{\log(1+(\beta-1)/\alpha)} \quad \text{--- (7)}$$

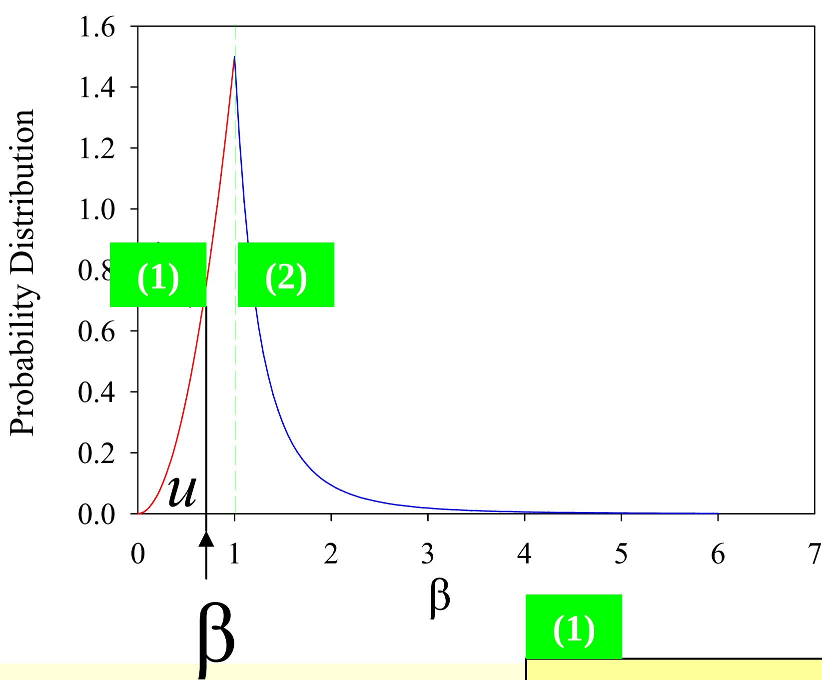
α is a constant ($\alpha > 1$)

Distribution Index (n_c) Update

So, we are done with the expanding case (2)

How about contracting case (1) ?

→ The same process is applied for (1)



(1) $c(\beta) = 0.5(n_c+1)\beta^{n_c}, \beta \le 1$

(2) $c(\beta) = 0.5(n_c+1)\frac{1}{\beta^{n_c+2}}, \beta > 1$

(1)

$$\int_0^{\beta} 0.5(n_c+1)\beta^{n_c} d\beta = u \quad \text{--- (x)}$$

Distribution Index (n_c) Update for contracting case

When c **is better than** p_1 and p_2 then the distribution index for the next generation is calculated as (8). The distribution index will be decreased in order to create an offspring (c') father away from its parent(p_2) than the offspring (c) in the previous generation.

$$\hat{n_c} = \frac{1+n_c}{\alpha} - 1 \qquad \text{--- (8)}$$

When c is worse than p_1 and p_2, $1/\alpha$ is used instead of α

$$\hat{n_c} = \alpha(1 + n_c) - 1 \qquad \text{--- (9)}$$

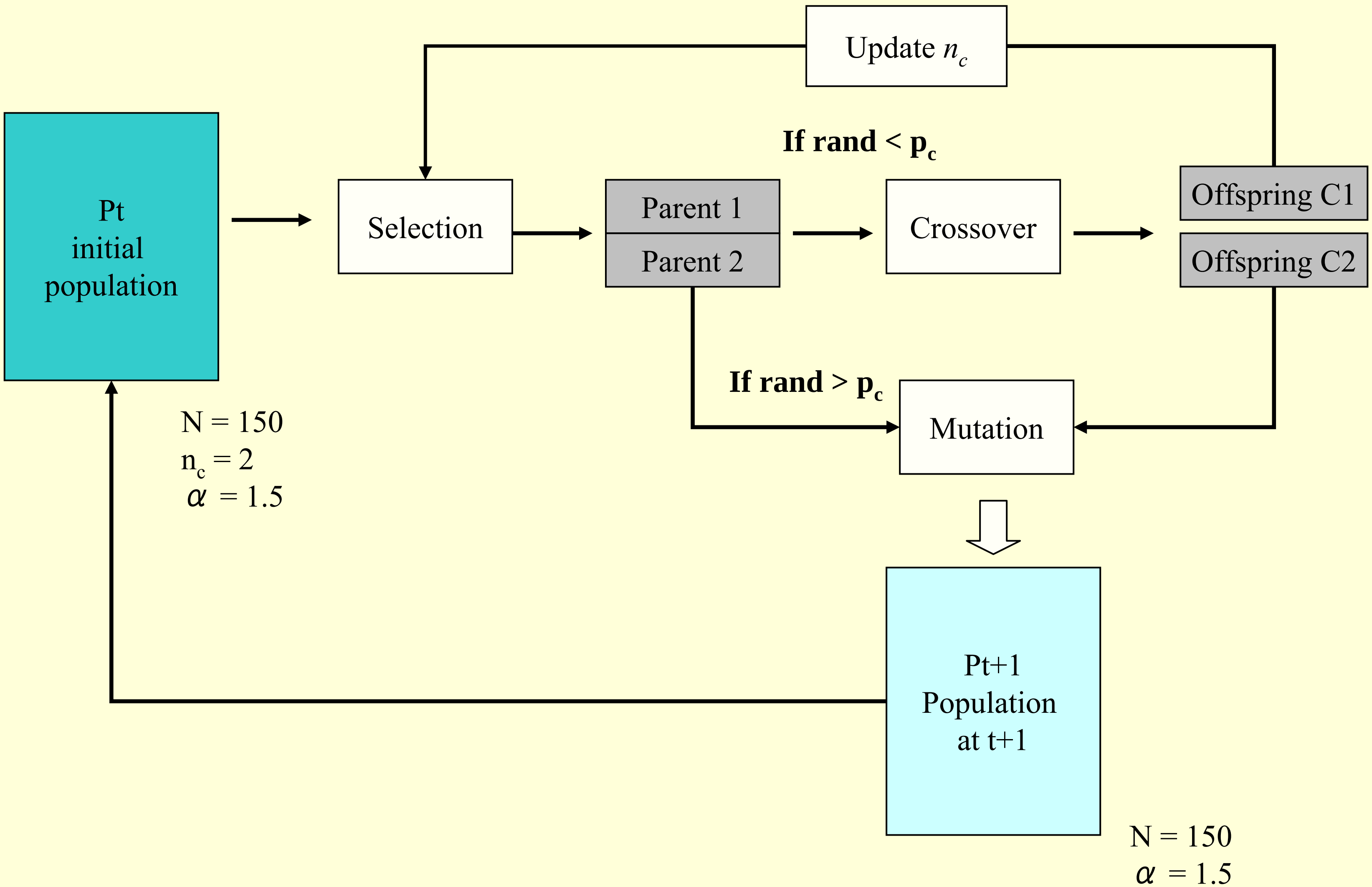
α is a constant ($\alpha > 1$)

Distribution Index Update Summary

	Expanding Case (u > 0.5)	Contracting Case (u<0.5)
c is better than parents	$\hat{n}_c = -1 + \frac{(n_c+1)\log\beta}{\log(1+\alpha(\beta-1))}$	$\hat{n}_c = \frac{1+n_c}{\alpha} - 1$
c is worse than parents	$\hat{n}_c = -1 + \frac{(n_c+1)\log\beta}{\log(1+(\beta-1)/\alpha)}$	$\hat{n}_c = \alpha(1+n_c) - 1$

(u = 0.5), $\hat{n}_c = n_c$

Self-SBX Main Loop



Simulation Results

Sphere Problem:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 .$$

SBX

- Initial n_c is 2.
- Population size =150
- Pc = 0.9
- The number of variables (n) =30
- Pm =1/30
- A run is terminated when a solution having a function value = 0.001
- 11 runs are applied.

Self-SBX

- Pc = 0.7
- α = 1.5

Fitness Value

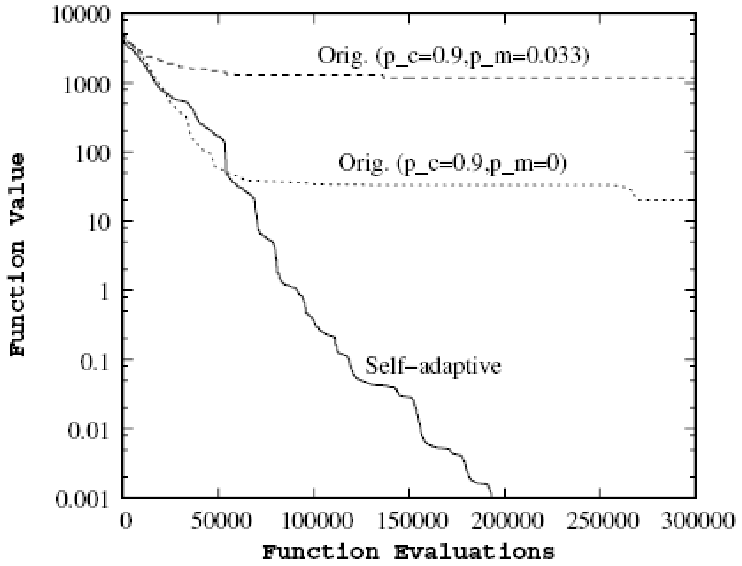


Figure 3: Variation of population-best function value with number of function evaluations for the 30-variable sphere function.

- Self-SBX can find better solutions

α Study

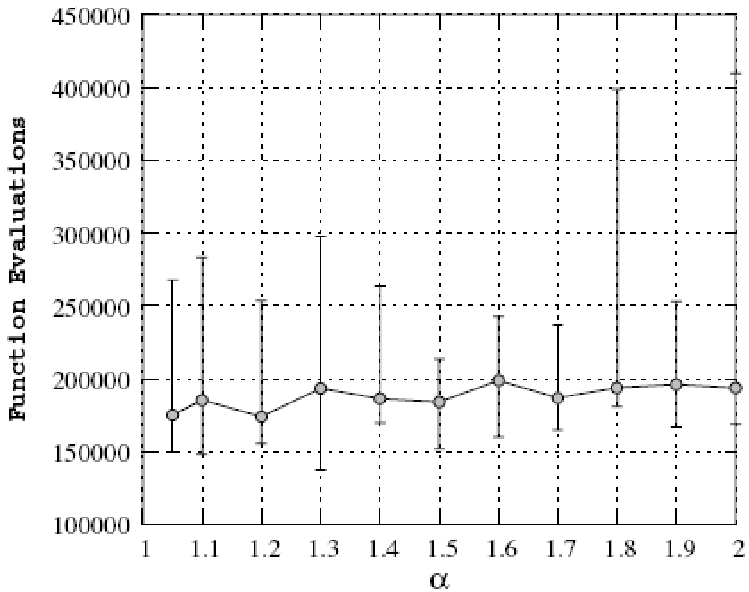


Figure 4: Parametric study of α for the 30-variable sphere function.

- α is varied in [1.05,2]
- The effect of α is not significant since the average number of function evaluations is similar.

Simulation Results

Rastrigin's Function:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + 10\left(1 - \cos(2\pi x_i)\right).$$

Many local optima, one global minimum (0)
The number of variables = 20

SBX

- Initial n_c is 2.
- Population size =100
- Pc = 0.9
- The number of variables (n) =30
- Pm =1/30
- A run is terminated when a solution having a function value = 0.001 or the maximum of 40,000 generations is reached.
- 11 runs are applied.

Self-SBX

- Pc = 0.7
- α = 1.5

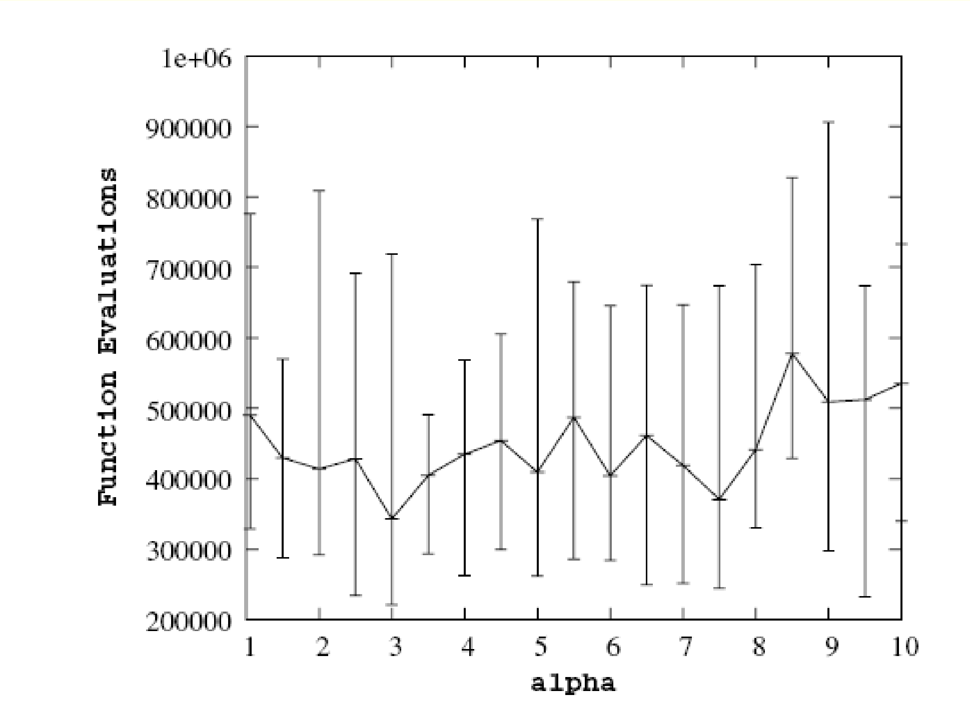
Fitness Value

Table 3: Performance of real-coded GAs with fixed and self-adaptive η_c update on 20-variable Rastrigin's function.

Method	Optimized function value (func. eval.)		
	Best	Median	Worst
Original ($p_c = 0.9$, $p_m = 0.05$)	319.523 (4M)	338.093 (4M)	342.363 (4M)
Original ($p_c = 0.7$, $p_m = 0.01$)	124.368 (4M)	210.929 (4M)	335.380 (4M)
Self-adp.($p_c = 0.7$, $p_m = 0.01$)	10^{-4} (287,822)	10^{-4} (429,511)	10^{-4} (569,597)

- Self-SBX can find a very close global optimal

α Study



- α is varied in [1.05,10]
- The effect of α is not significant since the average number of function evaluations is similar.
- α = 3 is the best

Self-SBX in MOOP

Table 4: Performance (hyper-volume) comparison of self-adaptive NSGA-II with fixed-η_c based SBX on ZDT1 and ZDT2.

α	Best	Median	Worst
ZDT1			
Original, fixed $\eta_c = 15$			
	0.72745	0.72133	0.72075
Self-Adaptive SBX			
1.05	0.72768	0.68531	0.62985
1.20	0.73949	0.72203	0.56250
1.50	0.72908	0.72222	0.61360
1.70	0.73984	0.72228	0.72210
2.00	0.72282	0.72238	0.72196
ZDT2			
Original, fixed $\eta_c = 15$			
	0.38883	0.38846	0.38800
Self-Adaptive SBX			
1.05	0.38959	0.37661	0.15350
1.20	0.38940	0.38912	0.38874
1.50	0.38957	0.38920	0.38891
1.70	0.39040	0.38930	0.38890
2.00	0.38942	0.38922	0.38880

- Self-SBX is applied in NSGA2-II.
- Larger hyper-volume is better

Conclusion

- Self-SBX is proposed in this paper.
  - The distribution index is adjust adaptively on the run time.
- The simulation results show that Self-SBX can find the better solutions in single objective optimization problems and multi-objective optimization problems.
- However, α is used to tune up the distribution index.