GA Operators

선택

```
Generate initial solutions (say, p solutions);
do {
        for (i=0; i<k; i++){
                Select two parent solutions;
                Crossover;
                Mutation;
        }
        Replace;
} while (not stopping condition);
Report the best solution;
```

Roulette-Wheel Selection, 가장 흔한 선택

- 각 해는 fitness 값 F_i에 비례하는 선택 기회를 갖는다
- $F_i = (C_w - C_i) + (C_w - C_b)/(k-1),\ k > 1$

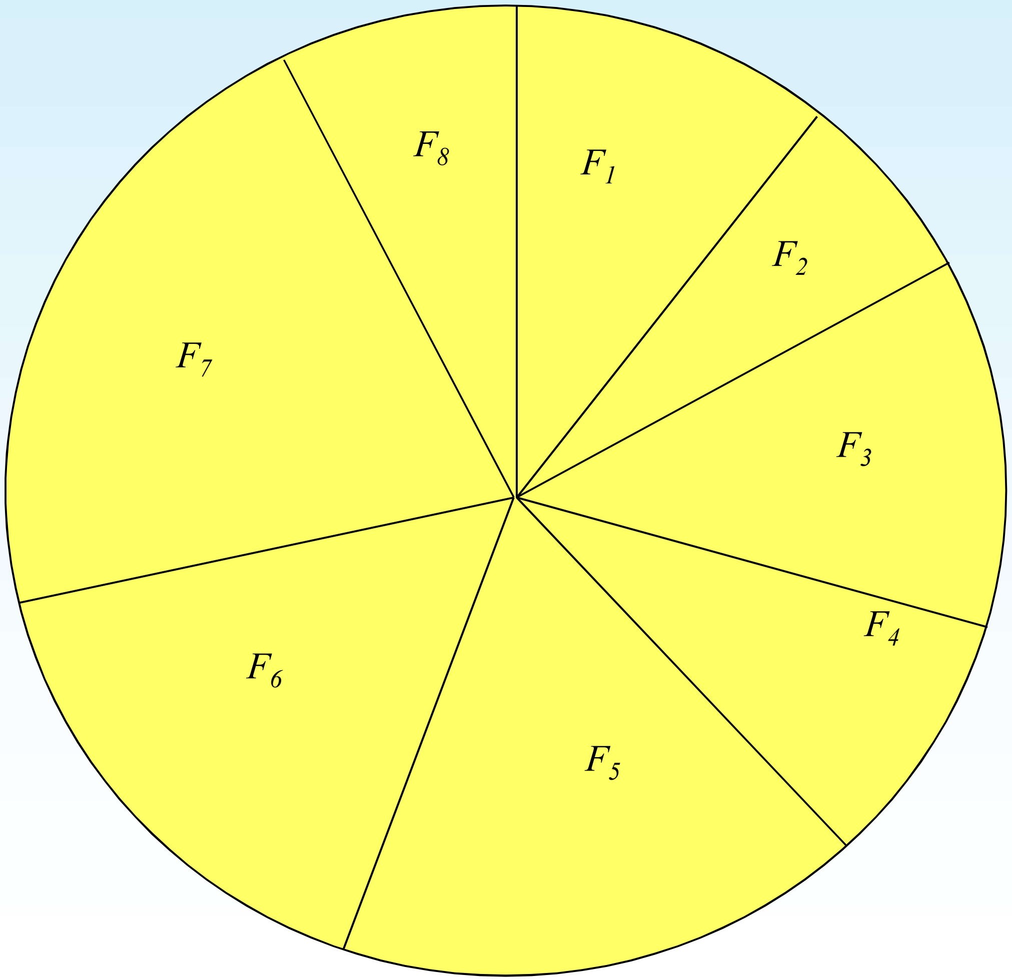
  Where

  C_w : population 내에서 가장 나쁜 해의 비용

  C_b : population 내에서 가장 좋은 해의 비용

  C_i: 해 i의 비용

사회에서 실패한 사람도 훌륭한 자식을 낳을 수 있다
단, 확률은 아무래도 낮다


F1
F2
F3
F4
F5
F6
F7
F8

구현은 간단하다

Selection

```
point ← random( ) % SumOfFitness;
sum ← 0;
for i ← 0 to P-1 {
    sum ← sum + f_i ;
    if (point < sum) then return i ;
}
Report error ; // shouldn't be here
```

Tournament Selection

- 변형들이 많음
- 대표적인 tournament selection

```
tSelect(t) {
    Choose two chromosomes x1 and x2; // Assume WLOG x1 is better
    r ← random number in [0, 1) ;
    if (t > r) then select x1 ;
               else  select x2 ;
}
```

- 다른 예

  Choose 2^k chromosomes ;

  Select a final one by tournament competition with the rule above ;

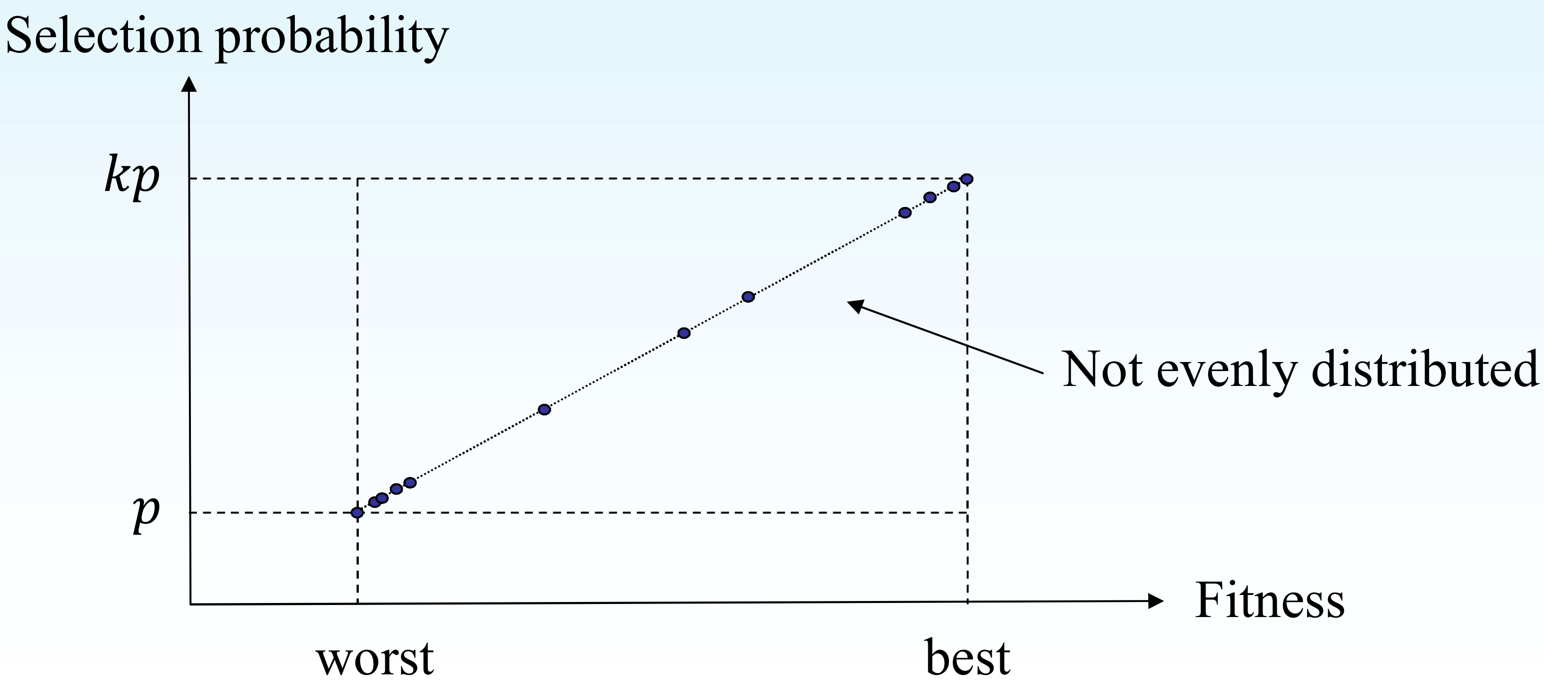
- What if $t = 1$?

  ✓ t로 selection pressure를 조절할 수도 있다

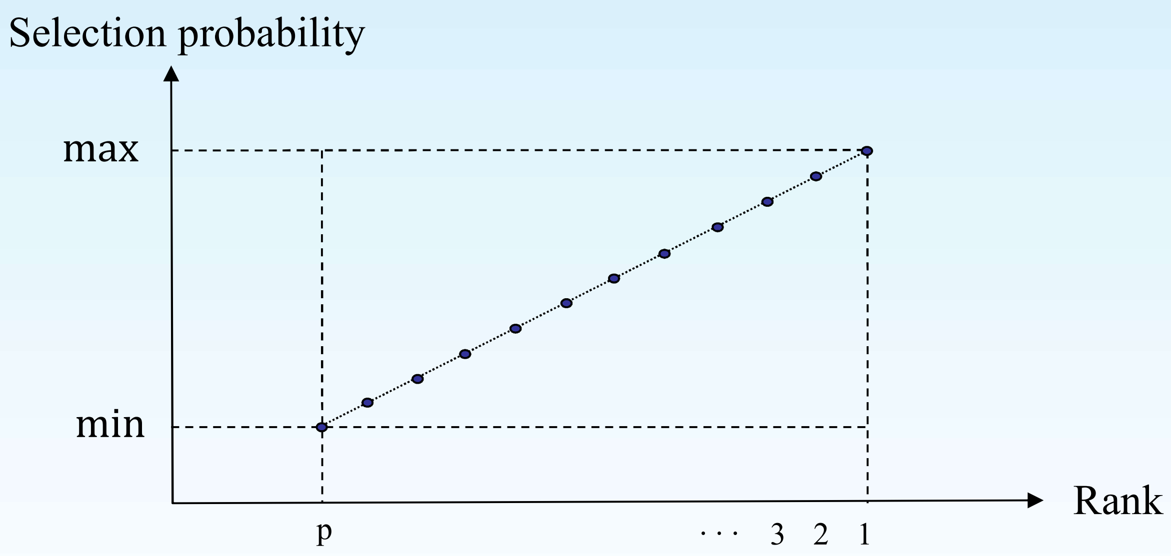
Rank-Based Selection

Roulette-wheel selection은 각 chromosome이 fitness에 비례하는 확률로 선택된다

- Roulette-wheel-based proportional selection이라고도 한다



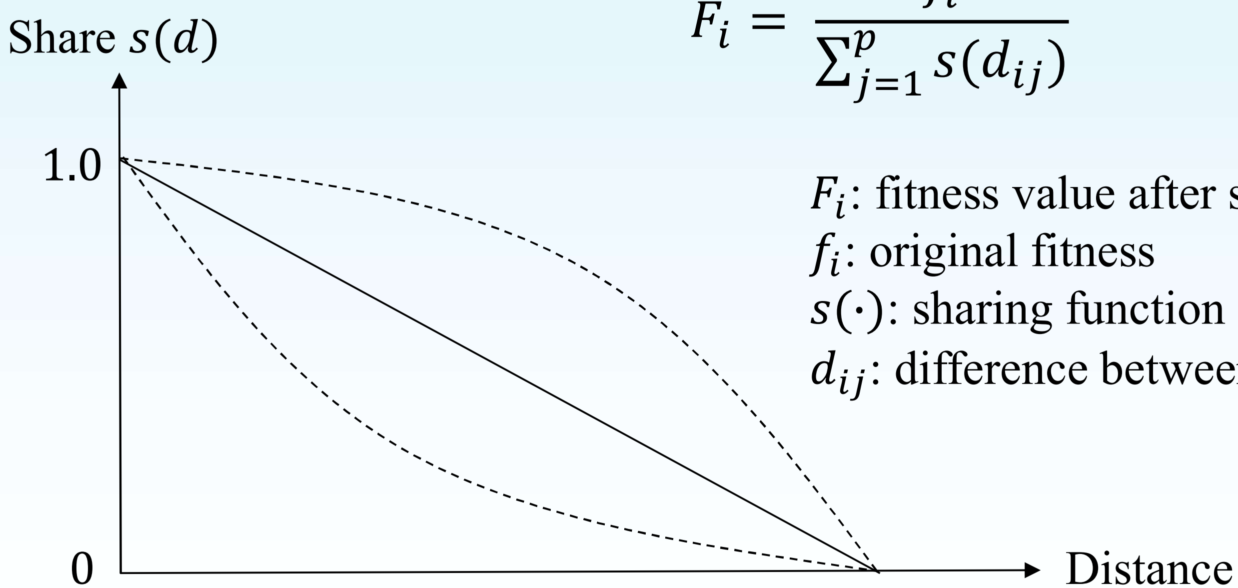
If we use “ranks” instead of fitness values, we can make the selection probabilities evenly distributed



- Rank i chromosome의 선택확률
  - $F_i = \max + (i-1)(\min - \max)/(P-1)$
- "$\min - \max$" can control the selection pressure
- But, even distribution is not a necessary property. We cannot say either of the two is better than the other.

Sharing

- Gives penalties to the chromosomes w/ many similar ones in the population
- Helpful to maintain population diversity



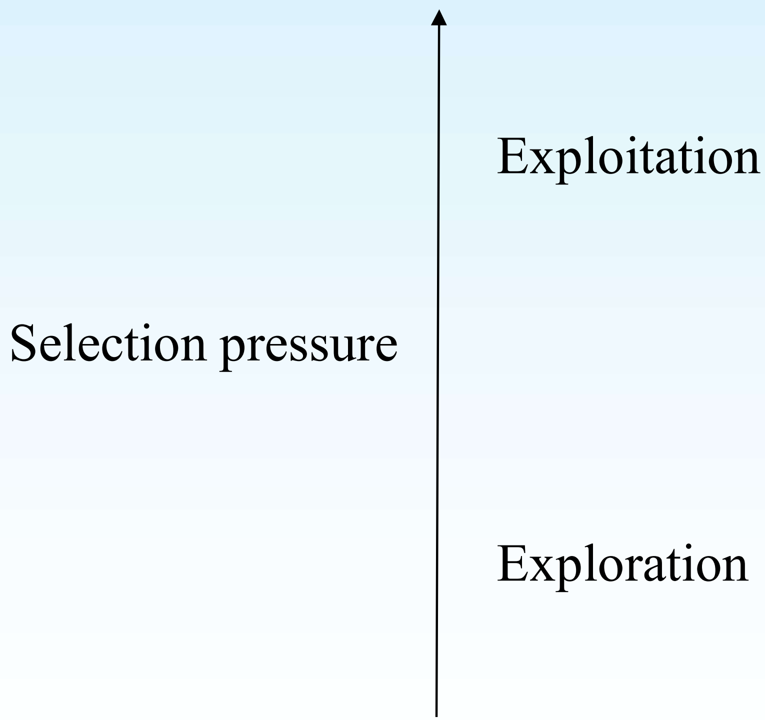
$d_{ij} = \|x_i - x_j\|$

$$F_i = \frac{f_i}{\sum_{j=1}^{p} s(d_{ij})}$$

F_i: fitness value after sharing
f_i: original fitness
$s(\cdot)$: sharing function
d_{ij}: difference between chromosomes i and j

Population Diversity & Selection Pressure

The increase of selection pressure decreases population diversity.



교차

```
Generate initial solutions (say, p solutions);
do {
        for (i=0; i<k; i++){
                Select two parent solutions;
                Crossover;
                Mutation;
        }
        Replace;
} while (not stopping condition);
Report the best solution;
```

1-Point Crossover

✓ GA의 대표 연산자

1-point crossover

```
a b c d e f | g h i j
k l m n o p | q r s t
---------------------
a b c d e f   q r s t
```

✓ There are $n - 1$ 1-pt xover operators for length-n chromosomes

Multi-Point Crossover

a b c d e f g h i j k l m

n o p q r s t u v w x y z

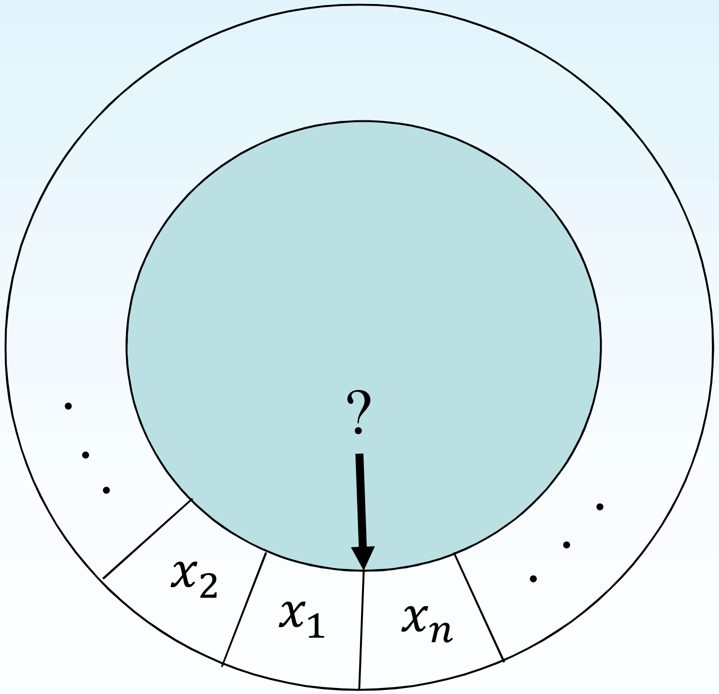
---

a b p q r f t u v w k l m

For $x_1 x_2 \cdots x_n$, if $n-1$ positions are allowed for cutting,
x_1 and x_n always move together in an "even"-point xover.

. . . apart in an "odd"-point xover.

Therefore, it is also reasonable to allow cutting after x_n



If we allow n-1 cutting positions,
there are $\binom{n-1}{k}$ k-point xover operators.

If we allow n cutting positions,
there are $\binom{n}{k}$ k-point xover operators.

- ✓ Multi-pt xovers are more useful in memetic GAs. (뒤에 소개함)

Uniform Crossover

$P_0 = 0.6$

s_1 :	a	b	c	d	e	f	g	h	i	j
s_2 :	k	l	m	n	o	p	q	r	s	t
난수 :	.83	.27	.33	.89	.91	.66	.44	.72	.42	.19
	a	l	m	d	e	f	q	h	s	t

- Uses no cutting points
- Generates a binary mask where 1's are uniformly distributed on the *n* positions
- $\exists 2^n$ masks(oparetors) $\gg \binom{n}{k}$
- Maximum perturbation occurs at $P_0 = 0.5$

Cycle Crossover

s_1 : 8 7 1 0 6 3 4 9 5 2

s_2 : 0 2 4 3 1 5 6 7 8 9

8 0 3 5

s_1 : 8 7 1 0 6 3 4 9 5 2

s_2 : 0 2 4 3 1 5 6 7 8 9

8 2 0 3 7 5 9

s_1 : 8 7 1 0 6 3 4 9 5 2

s_2 : 0 2 4 3 1 5 6 7 8 9

8 2 1 0 6 3 4 7 5 9

An important property of cycle xovers

- Every gene value in the offspring is from the same position of a parent

cf: PMX, Order xover

Order Crossover

s_1 : 8 7 1 0 6 3 4 9 5 2

s_2 : ~~0~~ 2 4 ~~3~~ 1 5 ~~6~~ 7 8 9

4 1 5 0 6 3 7 8 9 2

1. 임의로 2개의 cutting point를 정한다
2. 가운데 부분을 한 parent 1로부터 복사한다
3. parent 2에서 두번째 cutting point 다음부터 시작해서 중복을 피하면서 채워나간다

PMX (Partially Matched Crossover)

s_1 : 8 7 1 0 6 3 4 9 5 2

s_2 : 0 2 4 3 1 5 6 7 8 9

0 2 4 ~~3~~ 6 3 ~~6~~ 7 8 9

5 1

0 2 4 3 5 1 6 7 8 9

Arithmatic Crossover

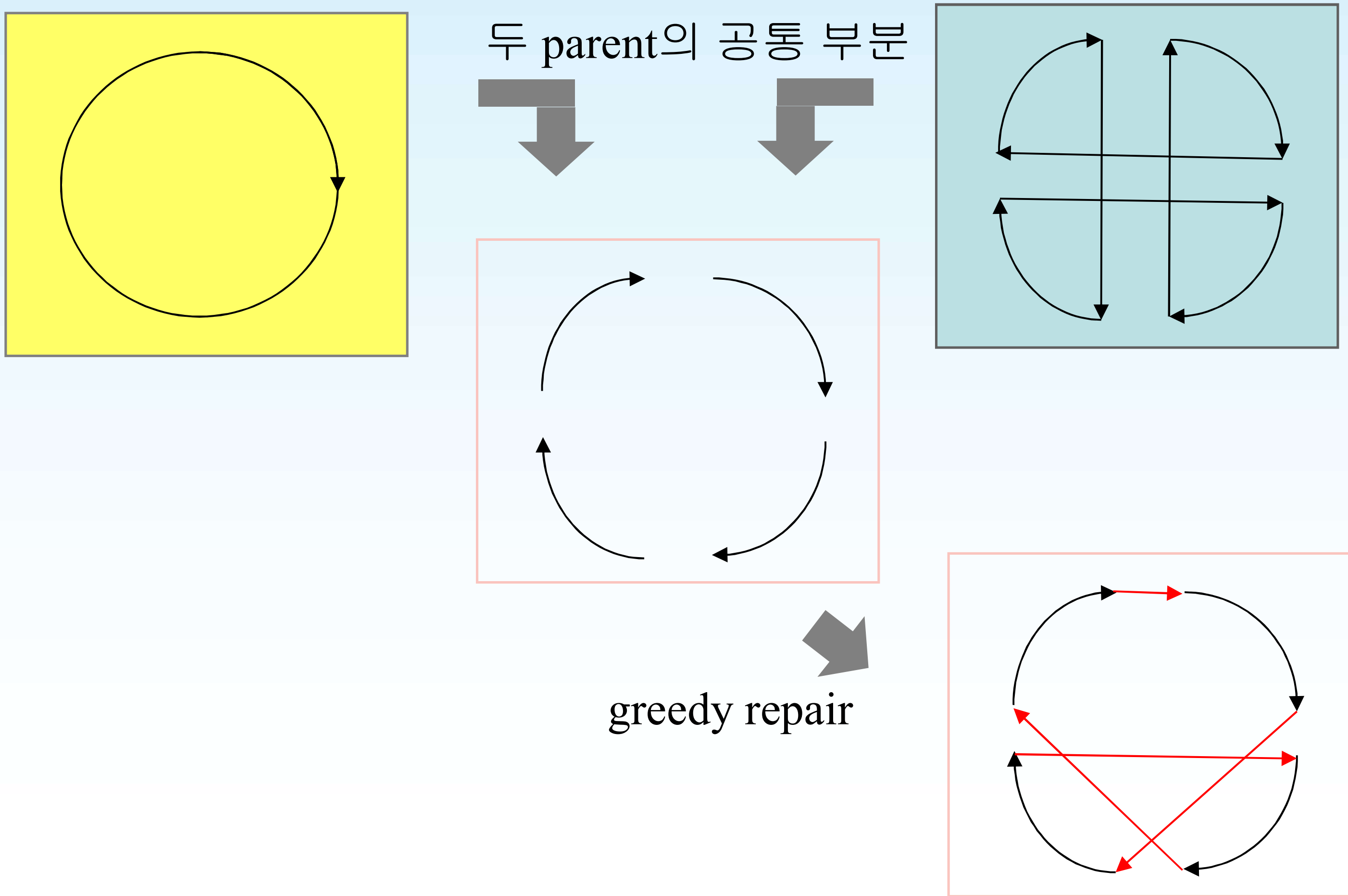
s_1 :	8	7.6	4.1	0	1.6	3.8	42
s_2 :	0	2.6	4.2	3	1.8	5.0	60
	4	5.1	4.25	1.5	1.7	4.4	51

각 유전자별로 평균을 취한다

Heuristic Crossovers

Crossovers that use problem-specific knowledge

예: DPX for TSP



Edge Recombination for TSP

s_1 : 0 1 2 3 4 5 6 8 7 9

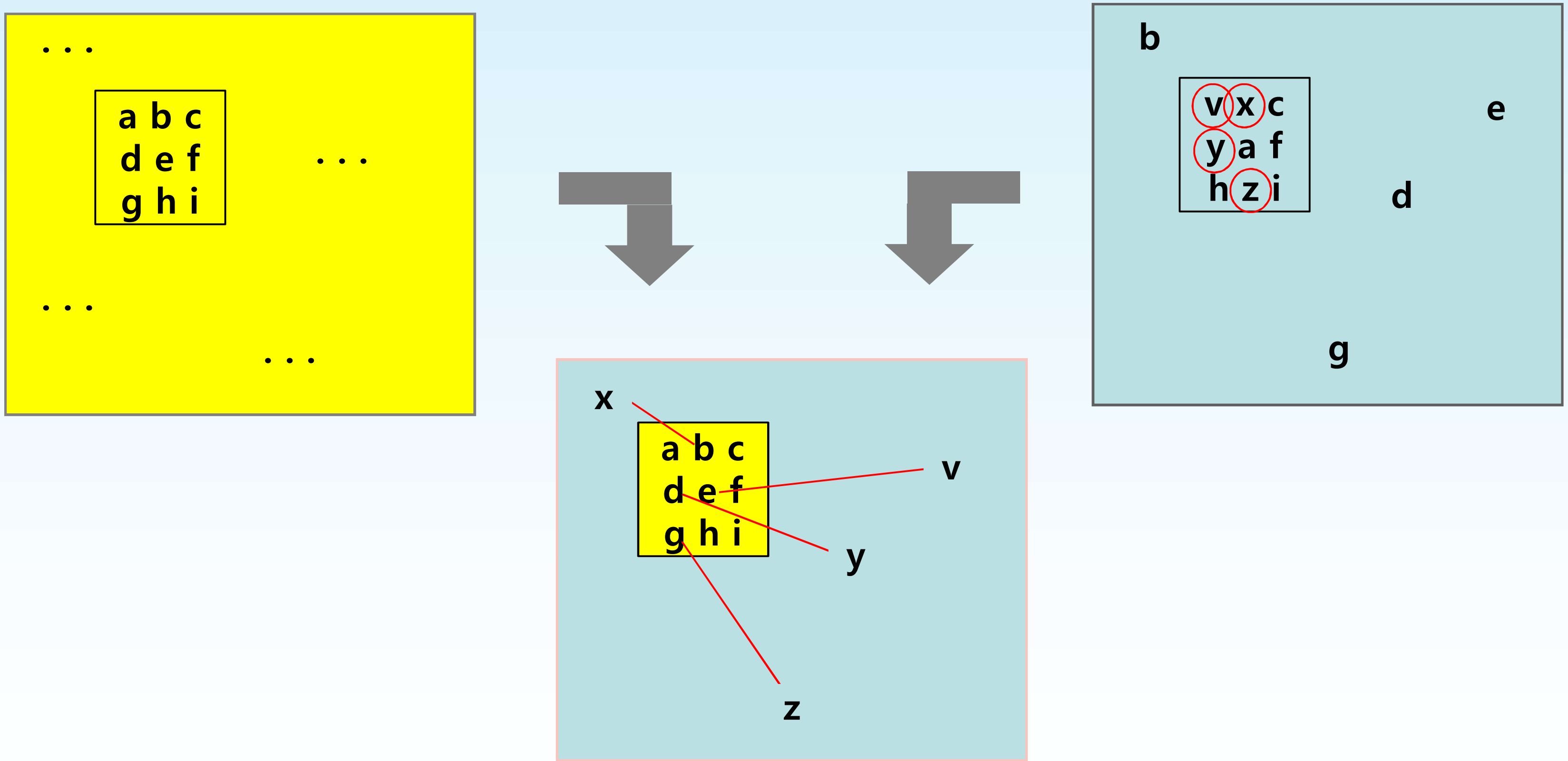
s_2 : 2 5 0 9 7 3 8 6 1 4

vertex	links	
0	9, 1, 5	
1	~~0~~, 2, ~~6~~, 4	
2	~~1~~, 3, ~~6~~, 4	
3	~~2~~, ~~4~~, ~~7~~, ~~8~~	5
4	3, 5, ~~1~~, ~~2~~	
5	4, 6, 2, 0	
6	5, ~~8~~, 1	
7	8, ~~9~~, 3	
8	6, ~~7~~, 3	
9	~~0~~, 7	

0 9 7 8 6 1 2 4 3 5

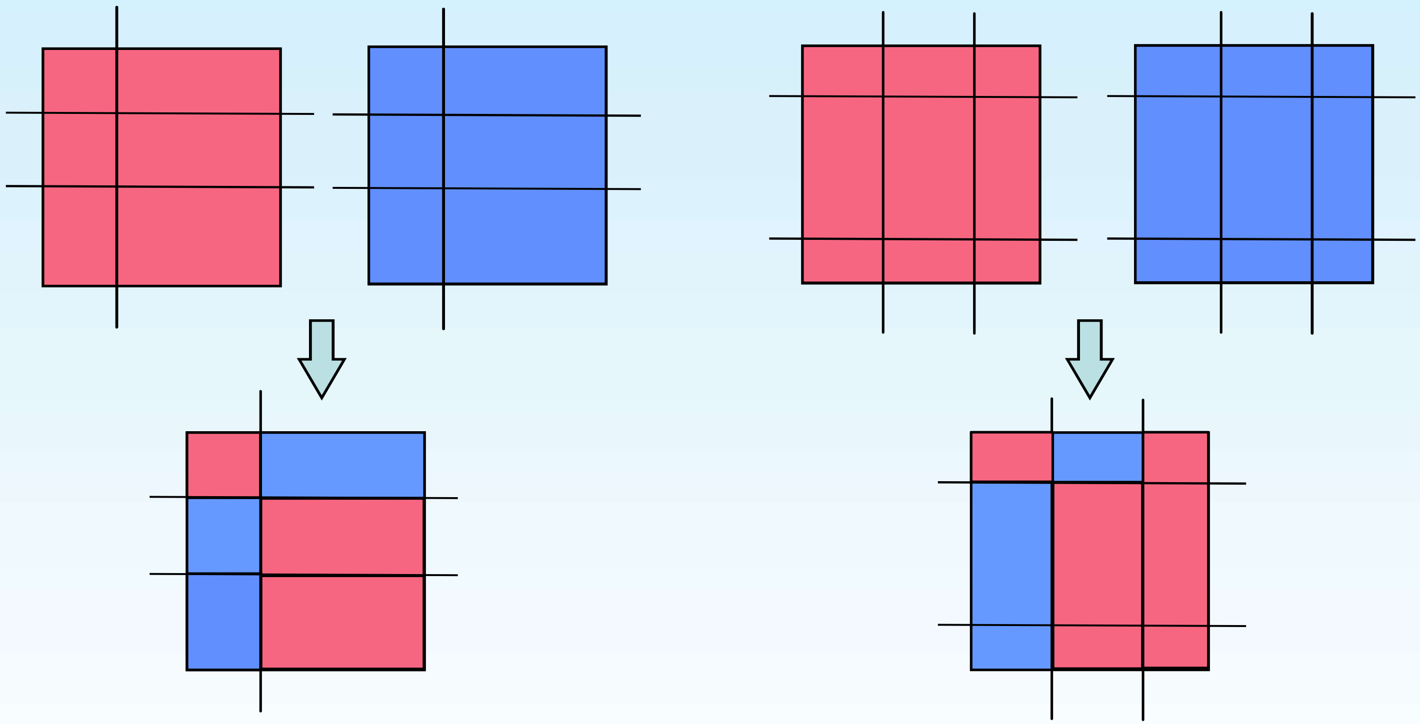
Cohoon&Paris' Crossover

VLSI-CAD Placement를 위해 고안
최초의 multi-dimensional xover



✓ 사실상 PMX를 2차원으로 확장한 것

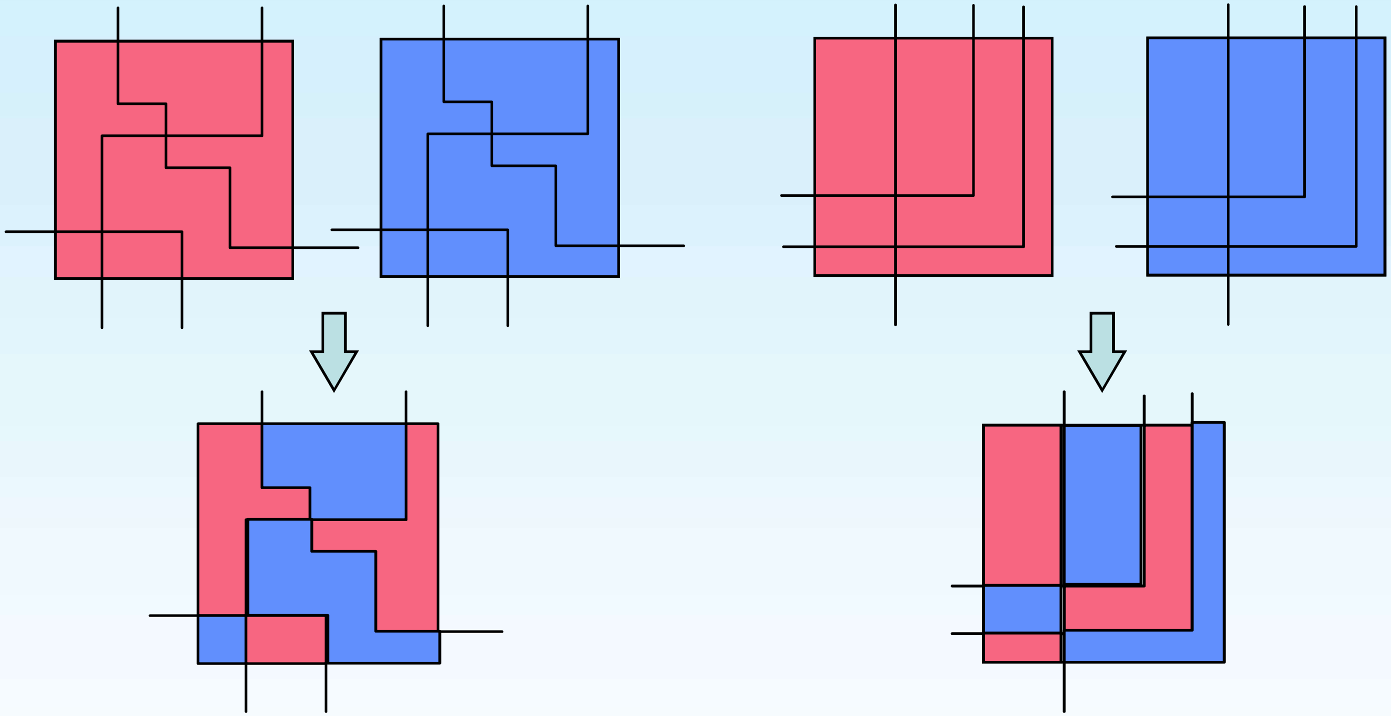
Block-Uniform Crossover



1. 직선을 사용하여 2차원 chromosome을 $i \times j$ 블록으로 분할한다
2. 각 블록마다 0 또는 1을 임의 할당한다 (uniform xover처럼)
3. 앞의 0 또는 1 값에 따라 각 블록을 각 parent로부터 복사한다

✓ 이것을 3차원 이상으로 확장하는 것은 어렵지 않다

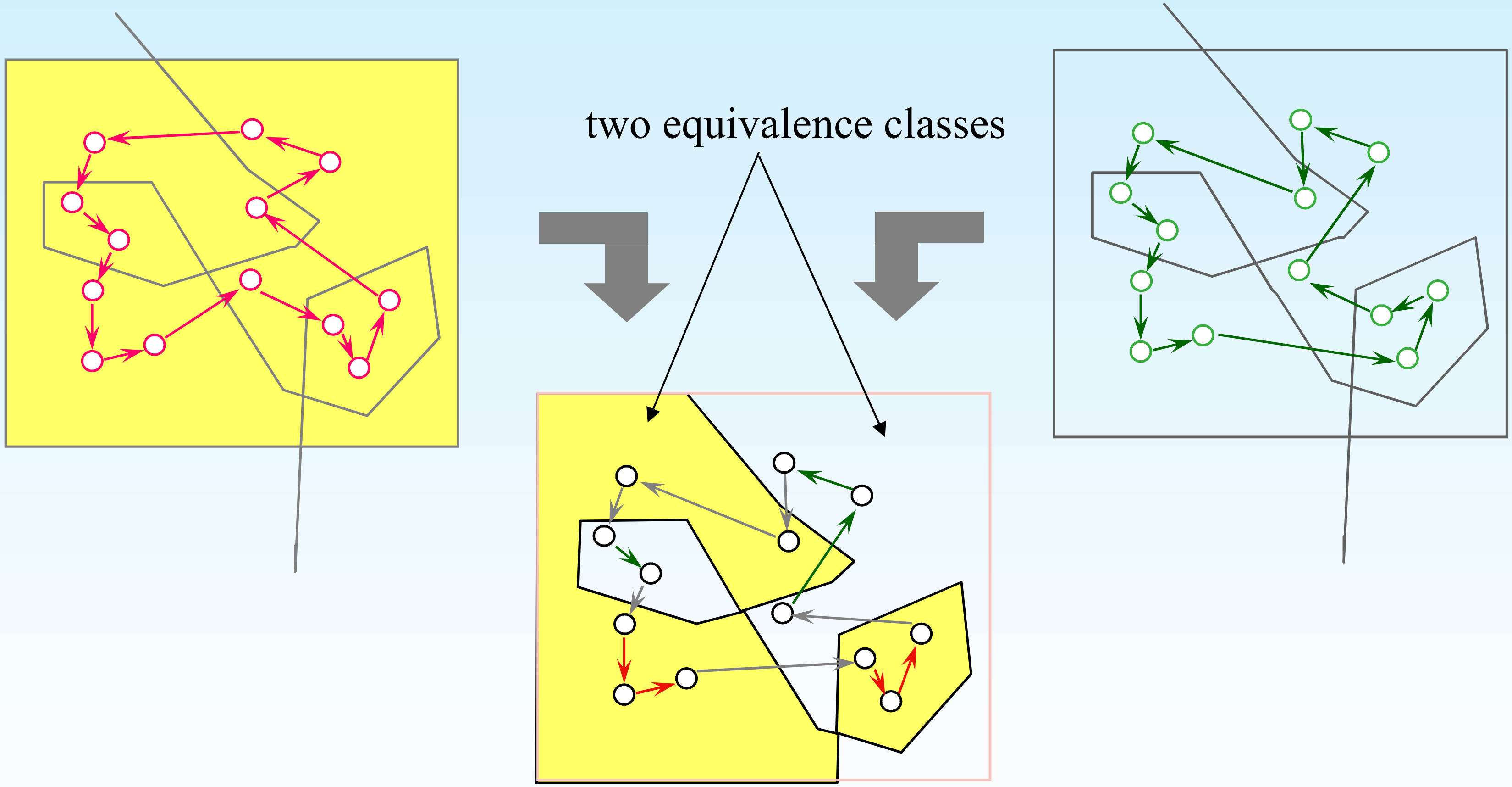
Geographic Crossover



1. 단조 직선을 사용하여 2차원 chromosome을 여러 블록으로 분할한다
   - 이렇게 분할된 블록들은 항상 2개의 equivalence class로 나눌 수 있다
2. 각 블록이 속한 equivalence class에 따라
   parent 1 또는 parent 2로부터 복사한다

✓ 교차 연산자의 다양성을 높인다

Natural Crossover for TSP



1. 2차원 chromosome에서 손의 궤적을 따라 k개의 cutting line을 그린다
2. 1의 결과로 모든 폐공간이 2개의 equivalence class로 나뉜다
3. 각 parent에서 자신의 equivalence class에 속하는 vertex들을 연결하는 edge들을 복사한다
4. 적당히 나머지 edge들을 만들어 Hamiltonian cycle을 만든다

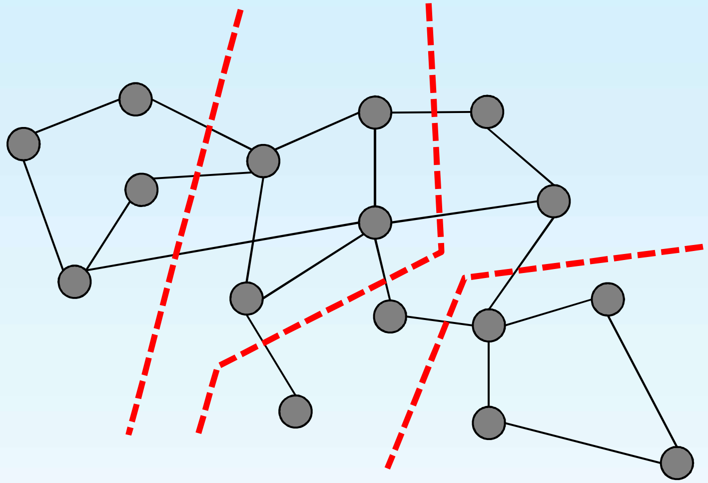
Normalization

e.g. 4-way partitioning problem

s_1 : 3 3 2 2 | 3 3 0 2 1 1 | 1 0 | 0 0 1 2

s_2 : 2 2 0 0 | 2 2 1 0 3 3 | 1 1 | 3 3 1 0

3 3 2 2 2 2 1 0 3 3 1 0 3 3 1 0



s_1'

2 2 0 0 | 2 2 3 0 1 1 | 1 3 | 3 3 1 0

normalization

s_1 : 3 3 2 2 | 3 3 0 2 1 1 | 1 0 | 0 0 1 2

s_2 : 2 2 0 0 | 2 2 1 0 3 3 | 1 1 | 3 3 1 0

2 2 0 0 2 2 1 0 3 3 1 3 3 3 1 0

$\|s_1' - s_2\| = 4$

$\|s_1 - s_2\| = 14$

Phenotype의 의미를 훨씬 살렸다

Phenotype은 비슷한데 genotype은 전혀 다르다

Normalization이 유용한 예

- Sorting networks
- neural networks
- graph partitioning
- number partitioning
- …

Proving Xovers' Validity

Alternate copy is clear in most cases.
But it is not clear in some cases particularly in multi-dimensional xovers.

Assume chromosomal domain space $D^n, D = \{0, 1, \ldots, l-1\}$.
(l-ary n-cube structure)

[Def. 1] A cutting surface in $R^n - D^n$ is a hypersurface
that divides the chromosomal domain space D^n
into exactly two disjoint subspaces

[Def. 2] For two points $x, y \in D^n$, $(x, y) \in R_e$ iff $\exists$ a path from x to y
that makes an even # of intersections with cutting surfaces

[Fact] Relation R_e is an equivalence relation

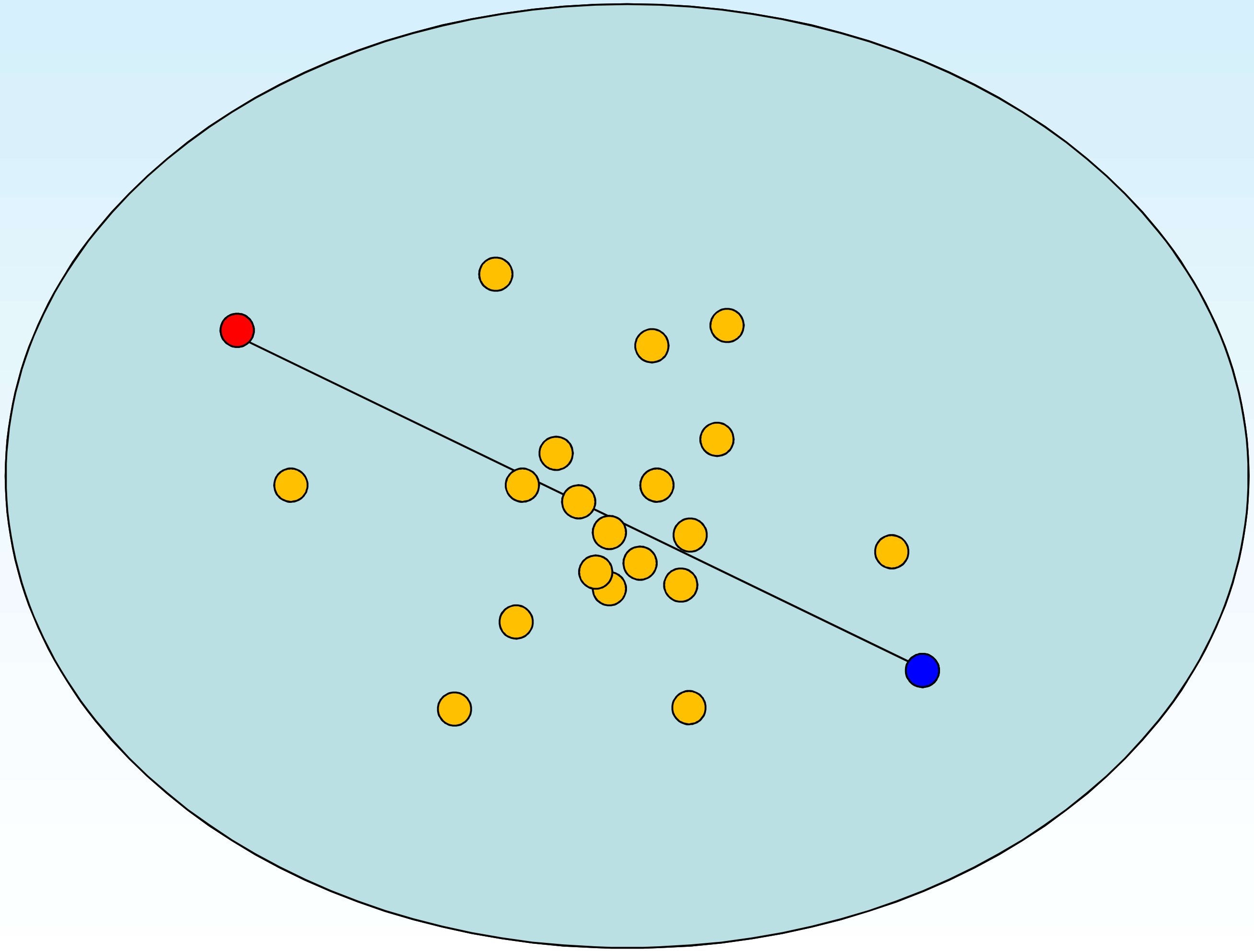
<Proof> $(x, x) \in R_e$ (reflexive)
$(x, y) \in R_e$ iff $(y, x) \in R_e$ (commutative)
$(x, y) \in R_e$ and $(y, z) \in R_e$ then $(x, z) \in R_e$ (transitive)

[Thm 1] If $(x, y) \in R_e$, then "any" path from x to y
also has an even # of intersections with cutting surfaces

[Thm 2] For any number k of cutting surfaces,
the equivalence relation R_e induces exactly two equivalence classes of points in D^n

✓ Thm 2 guarantees a valid crossover

공간탐색에서 교차의 역할



대략 두 chromosome의 중앙 부근을 지향한다

변이

```
Generate initial solutions (say, p solutions);
do {
        for (i=0; i<k; i++){
                Select two parent solutions;
                Crossover;
                Mutation;
        }
        Replace;
} while (not stopping condition);
Report the best solution;
```

변이

Perturbs a small fraction of an offspring to introduce unexplored solution space

Typical form

	a	b	c	d	e	f	g	h	I	j	k	
random #s :	0.25	0.22	0.03	0.9	0.71	0.45	0.63	0.87	0.2	0.99	0.16	$\in [0, 1]$
e.g. $P_t = 0.05$	a	b	c′	d	e	f	g	h	I	j	k	

Non-Uniform Mutation

The degree of perturbation decreases over time

$$v = \begin{cases} v + \Delta(t, \mathrm{UB} - v) & \text{if } r_1 = 0 \\ v + \Delta(t, v - LB) & \text{if } r_1 = 1 \end{cases}$$

gene value

where $r_1 \in \{0, 1\}$ (random)

$$\Delta(t, y) = y\left(1 - r_2^{\left(1 - \frac{t}{T}\right)^b}\right)$$

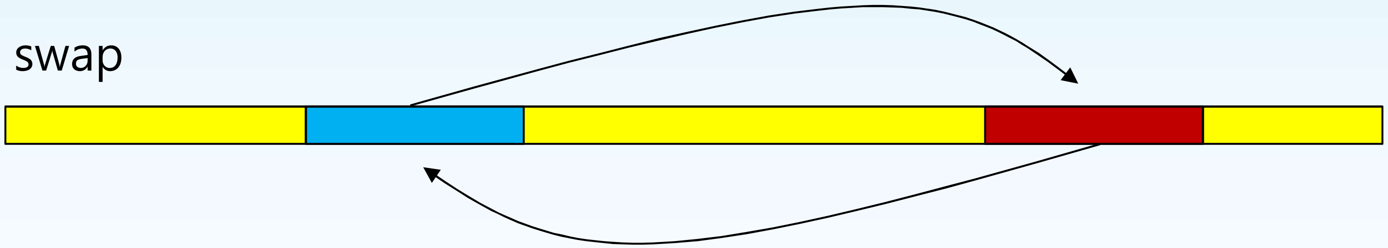
where $\begin{cases} r_2\text{: random number in } [0, 1] \\ T\text{: max generation} \\ b\text{: a parameter} \end{cases}$

- ✓ Okay in a pure GA
- ✓ But may not be good in a memetic GA

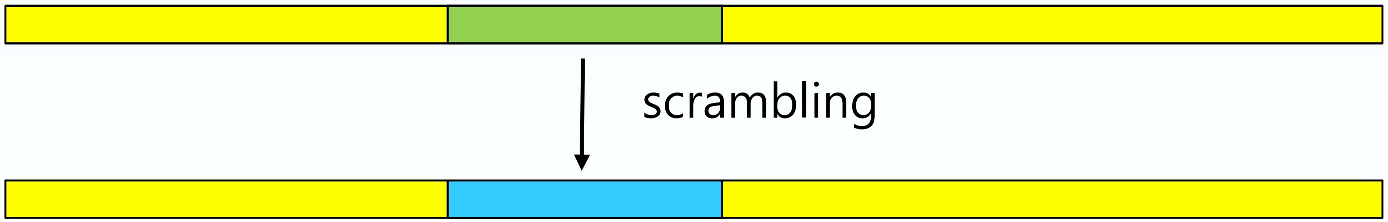
다른 변이들

Interval inversion

Interval swap

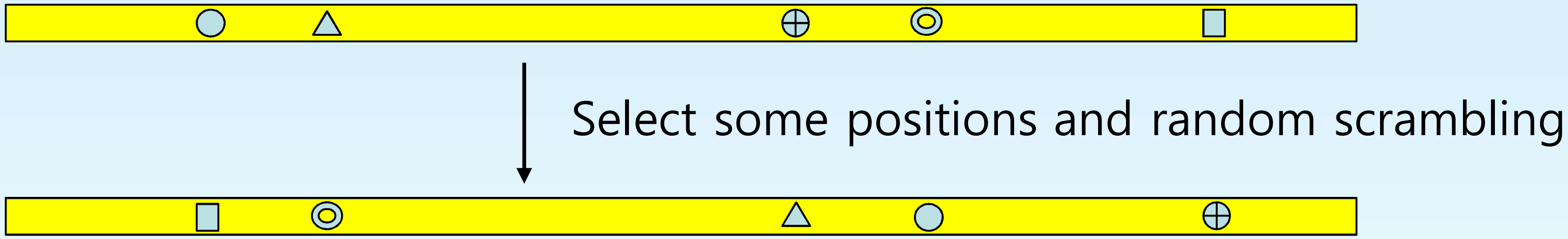


Interval scramble

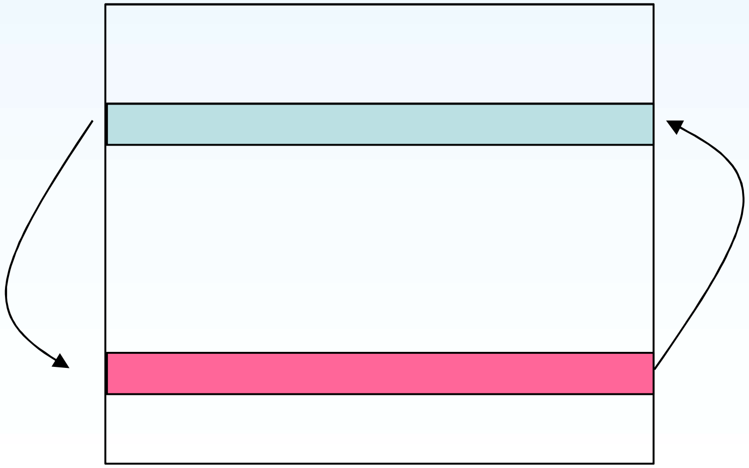


다른 변이들

Pick & scramble



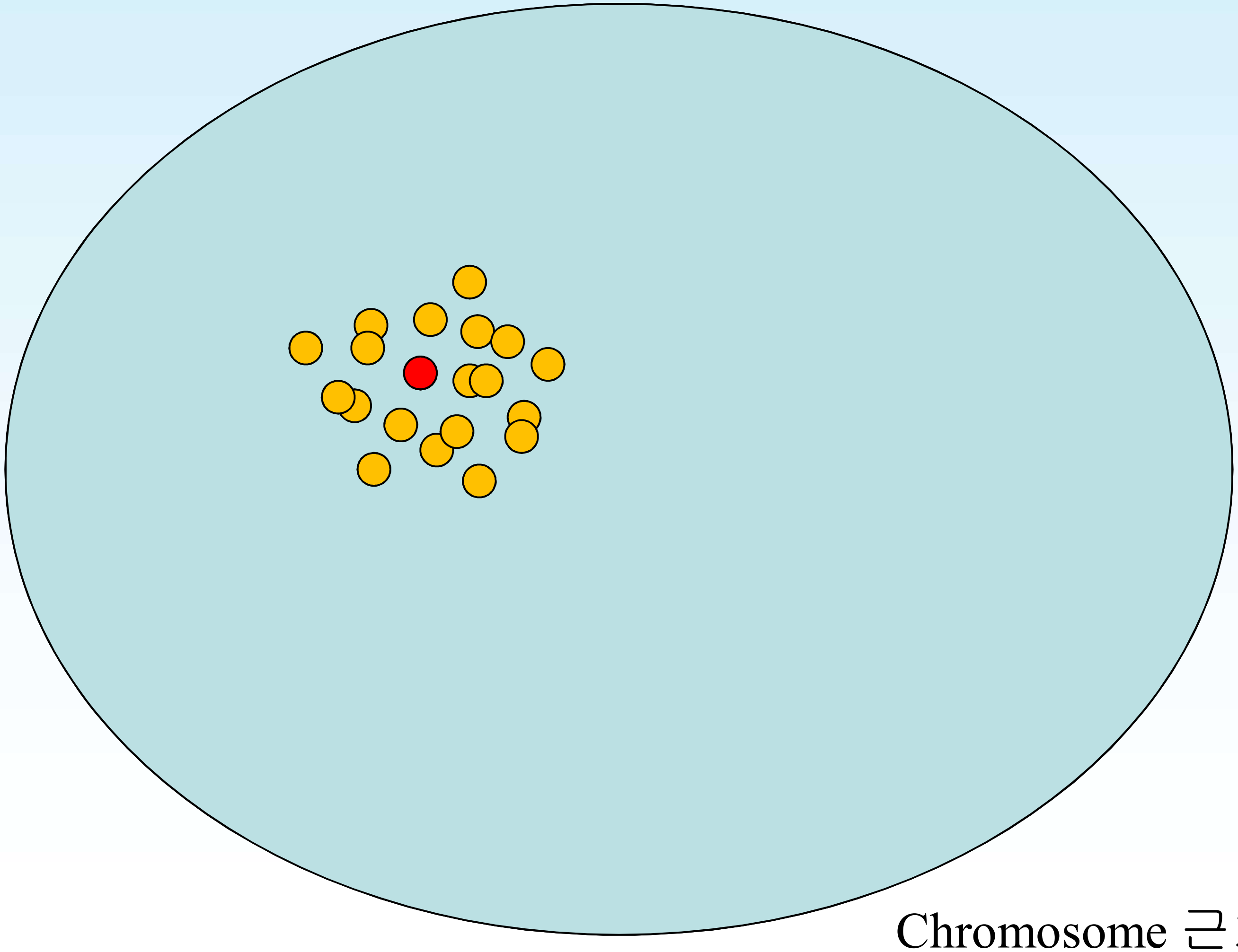
row swap



- Mutation can slow down the convergence.
- Mutation is a background operator in GA.
- But it is the main operator in ES and is the only operator in EP. (in the sense EP doesn’t use xover)
- Usually, steady-state GAs converge faster than generational GAs. Steady-state GAs are more probable to premature convergence. Stronger mutation can alleviate steady-state GAs’ premature convergence.

공간탐색에서 변이의 역할

대개, 해의 일부를 임의로 변형시킨다



Chromosome 근처에서 조금 옮긴다

대치

```
Generate initial solutions (say, p solutions);
do {
        for (i=0; i<k; i++){
                Select two parent solutions;
                Crossover;
                Mutation;
        }
        Replace;
} while (not stopping condition);
Report the best solution;
```

Replacement Strategies

✓ Easy in generational GAs (usually)

✓ There are some choices in steady-state GAs

- Genitor-style
  - Replace the worst solution in the population
- Preselection
  - Replace one of the parents
- Crowding
  - Randomly choose *k* solutions at ransom in the population
  - Replace the most similar one to the offspring
- Combinations