Lect12 Genetic Algorithm

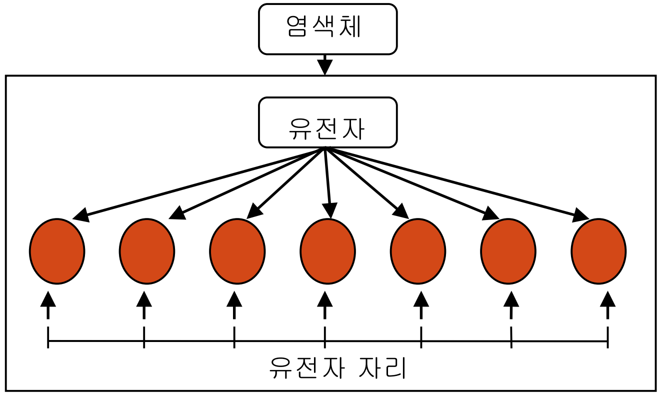
Genetic Algorithm

- 1962년 미국 미시간 대학의 John Holland 교수가 세포의 작용을 연구하던 중 제안한 것으로, 생물학계의 적자생존 원리 즉, 자연도태 원리를 기초로 한 최적화 알고리즘. 1975년 Adaptation in Natural and Artificial Systems 란 저서로 출판
- 생물의 유전과 진화알고리즘을 공학적으로 모델화하여 문제해결이나 시스템의 학습등에 응용하려는 알고리즘
- 어떤 세대(generation)을 형성하는 개체(individual)들의 집합, 즉 개체군(population)중에서 환경에 대한 적합도(fitness)가 높은 개체가 높은 확률로 살아남아 재생(reproduction)할 수 있게 되며 이때 교배(crossover) 및 돌연변이(mutation)로서 다음 세대의 개체군을 형성하게 되는 생물의 진화과정을 인공적으로 모델링한 알고리즘
- 진화적 알고리즘(evolutionary algorithm)이라고도 함
- 1992년, John Koza 는 "genetic programming" (GP) 기법으로 GA를 구현
- 2000년대 붐을 일으키고 있으며, Job shop scheduling, 훈련신경회로망, 이미지 특징 추출 및 인식, 최적화 문제등에 응용

Genetic Algorithm이란?

- Genetic Algorithm(GA)는 최적화 및 검색 문제에 대한 해 또는 근사해를 찾기 위해 컴
- GA는 휴리스틱 global 탐색기법으로 분류
- GA는 상속(inheritance), 돌연변이(mutation), 선택(selection) 및 교차(crossover) (재조합(recombination)이라고도 함)와 같은 진화 생물학에서 영감을 얻은 기술을 사용하는 진화 알고리즘(evolutionary algorithm)의 특정 클래스.
- 진화는 대개 무작위로 생성 된 개체의 개체군에서부터 시작되며 여러 세대에 걸쳐 발생
- 각 세대마다 개체군의 모든 개체의 적합성이 평가되고, 현재의 개체군 (건강 상태에 따라)에서 여러 개체가 선택되고 새로운 개체군을 형성하도록 수정됨.
- 새로운 개체군은 알고리즘의 다음 iteration에서 사용
- 알고리즘은 최대 수의 세대가 생성되거나 개체군에 대해 만족스러운 적합성 수준에 도달하면 종료

생물학과 GA 용어비교



<염색체, 유전자, 유전자자리>

생물학	유전자 알고리즘(GA)
개체(individual)	염색체에 의해 특징지어지는 자율적인 하나의 작은 집단
집단 (population)	집단 내의 개체의 수
유전자(gene)	개체의 형질을 규정하는 기본 구성요소 즉, 특성(feature), 형질(character) 등
염색체 (chromosome)	복수의 유전자 모임. 문자열(string)로 표현
대립 유전자(allele)	유전자가 갖는 특성 값(feature value).
유전자 자리(locus)	염색체상의 유전자의 위치 즉, 문자열의 위치 (string position)
적합도 또는 적응도(fitness)	유전자의 각 개체의 환경에 대한 적합의 비율을 평가하는 값 즉, 평가치로 최적화 문제를 대상으로 하는 경우 목적함수 값이나 제약조건을 고려하여 페널티 함수 값의 적응도로 설정된다.
코딩(coding)	표현 디코딩에서 유전자형으로 매핑하는 것
디코딩 (decoding)	유전자 형에서 표현형으로 역 매핑하는 것
유전자형 (genotype)	형질의 염색체에 의한 내부적으로 표현하는 방법으로 구조체(structure)로 표현
표현형 (phenotype)	염색체에 의해 규정된 형질을 외부적으로 표현하는 방법으로 파라메터 집합(parameter set), 대체해(alternative solution),디코드화를 위한 구조체(decoded structure)로 표현

Genetic Algorithm

produce an initial population of individuals

evaluate the fitness of all individuals

while termination condition not met **do**

    select fitter individuals for reproduction

    recombine between individuals

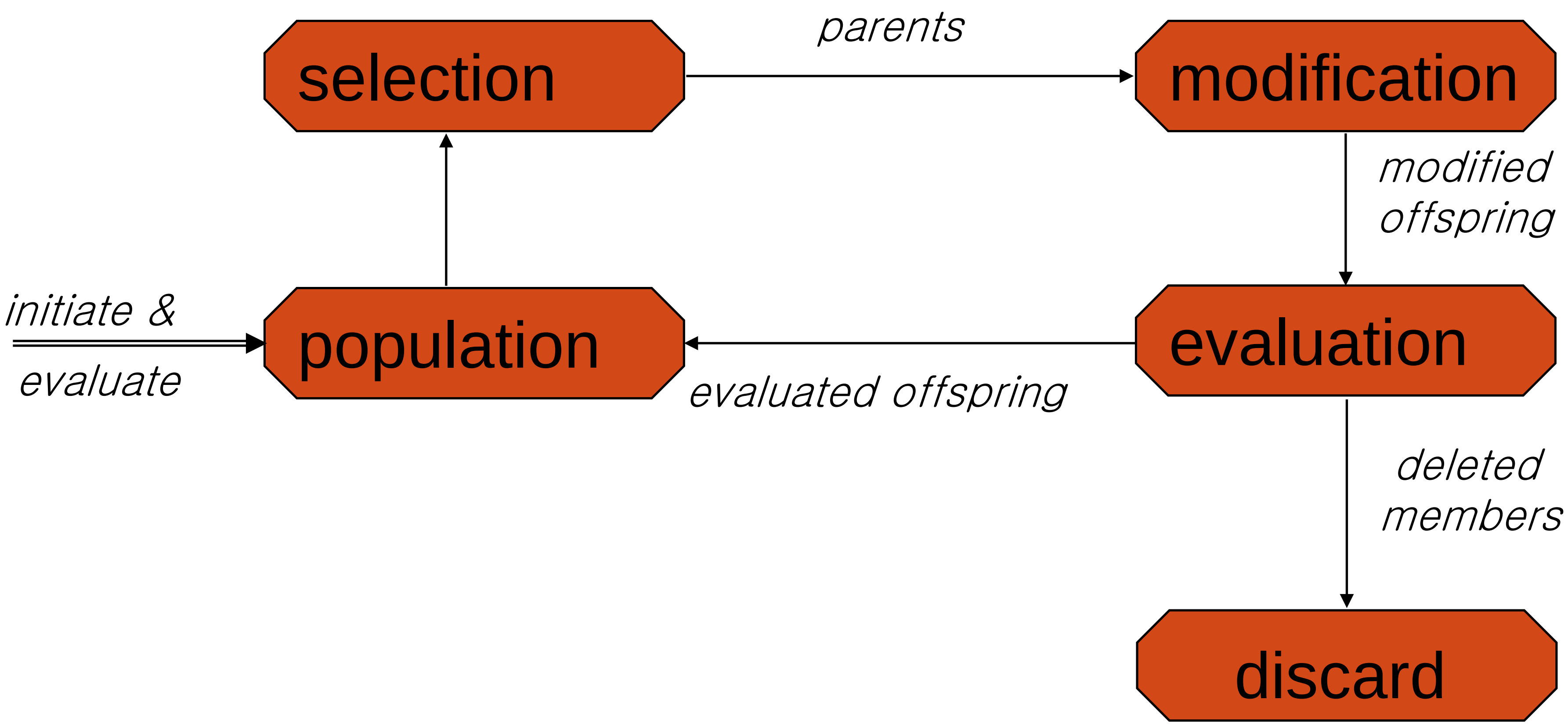
    mutate individuals

    evaluate the fitness of the modified individuals

    generate a new population

End while

The Evolutionary Cycle



Ex. Simple problem to understand genetic algorithm

- To find the value of x that maximizes

$$f(x) = \sin(\frac{x\,\pi}{256}),\ 0 \le x \le 255$$

1. Choose an alphabet to represent solutions to the problem. Ex. 8 bit for individuals. 10111101 (189)
2. Decide how many individuals make up a population. Ex. 8 individuals
3. Decide how to initialize the population. Ex. Randomly … (table)
4. Decide how to evaluate fitness. Ex. Use function.

Individual
0 1 1 0 0 0 1 1
0 0 1 1 0 1 0 1
1 1 0 1 1 0 0 0
1 0 1 0 1 1 1 0
0 1 0 0 1 0 1 0
1 0 1 0 1 1 1 0
0 1 1 0 0 0 1 1
1 0 1 1 1 1 0 1

Ex – f(x)

5. Decide which individuals to select for reproduction. Exploration with exploitation.
   - Make roulette wheel for each individual based on fitness. F -> normalized f -> assign roulette wheel -> random number -> select (Tale 10.1)
6. Determine how to perform crossover and mutations.
7. Decide when to terminate

Table 10.1 Initial Population of Individuals and Their Fitnesses

Individual	x	$f(x)$	Normed $f(x)$	Cumulative Normed $f(x)$
1 0 1 1 1 1 0 1	189	.733	.144	.144
1 1 0 1 1 0 0 0	216	.471	.093	.237
0 1 1 0 0 0 1 1	99	.937	.184	.421
1 1 1 0 1 1 0 0	236	.243	.048	.469
1 0 1 0 1 1 1 0	174	.845	.166	.635
0 0 1 0 0 0 1 1	74	.788	.155	.790
0 0 1 0 0 0 1 1	35	.416	.082	.872
0 0 1 1 0 1 0 1	53	.650	.128	1.000

Crossover

- Table 10.3 Parents and Children Resulting from Crossover

Parents	Children	x	$f(x)$
0 1 1^1 0 0 0 ^{2}1 1	0 1 1^1 1 0 1 ^{2}1 1	119	.994
0 0 1 1 0 1 0 1	0 0 1 0 0 0 0 1	33	.394
1^1 1 0 1 1 ^{2}0 0 0	1^1 0 1 0 1 ^{2}0 0 0	168	.882
1 0 1 0 1 1 1 0	1 1 0 1 1 1 1 0	222	.405
0 1 ^{2}0 0 1 0 1^1 0	1 0 ^{2}0 0 1 0 1^1 0	138	.992
1 0 1 0 1 1 1 0	0 1 1 0 1 1 1 0	110	.976
0 1 1 0 0^1 0 1 1 ^{2}	0 1 1 0 0^1 1 0 1 ^{2}	101	.946
1 0 1 1 1 1 0 1	1 0 1 1 1 0 1 1	187	.749

Example – MAXONE problem : to understand GA

- Suppose we want to maximize the number of ones in a string of *l* binary digits
- An individual is encoded (naturally) as a string of *l* binary digits
- The fitness *f* of a candidate solution to the MAXONE problem is the number of ones in its genetic code
- We start with a population of *n* random strings. Suppose that $l = 10$ and $n = 6$
- We toss a fair coin 60 times and get the following initial population:

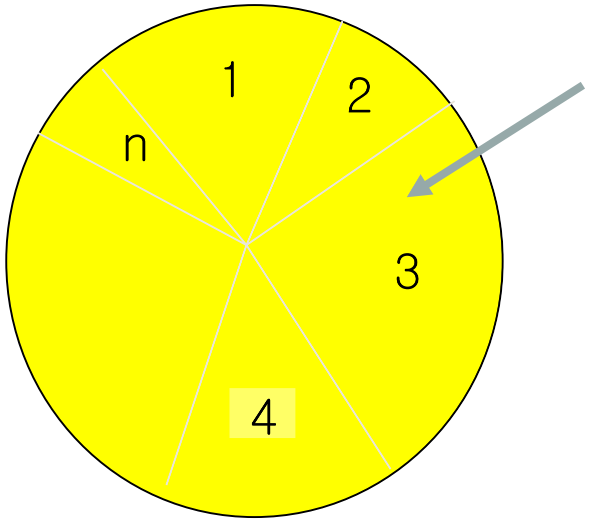
$$
\begin{array}{ll}
s_1 = 1111010101 & f(s_1) = 7 \\
s_2 = 0111000101 & f(s_2) = 5 \\
s_3 = 1110110101 & f(s_3) = 7 \\
s_4 = 0100010011 & f(s_4) = 4 \\
s_5 = 1110111101 & f(s_5) = 8 \\
s_6 = 0100110000 & f(s_6) = 3
\end{array}
$$

Example – MAXONE problem : to understand GA(2)

- Step 1 : Selection
  - We randomly (using a biased coin) select a subset of the individuals based on their fitness:

Individual *i* will have a $\frac{f(i)}{\sum_i f(i)}$ probability to be chosen

We repeat the extraction as many times as the number of individuals we need to have the same parent population size (6 in our case)



Area is Proportional to fitness value

Example – MAXONE problem : to understand GA(3)

- Selection set
  - Suppose that, after performing selection, we get the following population:

$s_1` = 1111010101$ (s_1)

$s_2` = 1110110101$ (s_3)

$s_3` = 1110111101$ (s_5)

$s_4` = 0111000101$ (s_2)

$s_5` = 0100010011$ (s_4)

$s_6` = 1110111101$ (s_5)

Example – MAXONE problem : to understand GA(4)

- Step 2 : crossover

  - Next we mate strings for crossover. For each couple we first decide (using some pre-defined probability, for instance 0.6) whether to actually perform the crossover or not
  - Suppose that we decide to actually perform crossover only for couples (s_1\`, $s_2$\`) and (s_5\`, $s_6$\`). For each couple, we randomly extract the crossover points, for instance 2 and 5

Example – MAXONE problem : to understand GA(5)

- Crossover result
  - Before crossover:

$s_1' = 1111010101$

$s_2' = 1110110101$

$s_5' = 0100010011$

$s_6' = 1110111101$

  - After crossover:

$s_1'' = 1110110101$

$s_2'' = 1111010101$

$s_5'' = 0100011101$

$s_6'' = 1110110011$

Example – MAXONE problem : to understand GA(6)

- Step3:Mutation
  - The final step is to apply random mutation: for each bit that we are to copy to the new population we allow a small probability of error (for instance 0.1)
  - Before applying mutation:

$$s_1`` = 1110110101$$

$$s_2`` = 1111010101$$

$$s_3`` = 1110111101$$

$$s_4`` = 0111000101$$

$$s_5`` = 0100011101$$

$$s_6`` = 1110110011$$

Example – MAXONE problem : to understand GA(7)

- After applying mutation
  - :

$$s_1``` = 1110100101 \quad f(s_1```) = 6$$

$$s_2``` = 1111110100 \quad f(s_2```) = 7$$

$$s_3``` = 1110101111 \quad f(s_3```) = 8$$

$$s_4``` = 0111000101 \quad f(s_4```) = 5$$

$$s_5``` = 0100011101 \quad f(s_5```) = 5$$

$$s_6``` = 1110110001 \quad f(s_6```) = 6$$

Example – MAXONE problem : to understand GA(8)

- In one generation, the total population fitness changed from 34 to 37, thus improved by ~9%

- At this point, we go through the same process all over again, until a stopping criterion is met

Components of a GA

A problem definition as input, and

- Encoding principles (gene, chromosome)
- Initialization procedure (creation)
- Selection of parents (reproduction)
- Genetic operators (mutation, recombination)
- Evaluation function (environment)
- Termination condition

Population

population

Chromosomes could be:

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

Encoding

- *The process of representing the solution in the form of a **string** that conveys the necessary information.*

- Just as in a chromosome, each gene controls a particular characteristic of the individual, similarly, each element in the string represents a characteristic of the solution.

Encoding Methods

- **Binary Encoding** – Most common method of encoding. Chromosomes are strings of 1s and 0s and each position in the chromosome represents a particular characteristic of the problem.

Chromosome A	10110010110011100101
Chromosome B	11111110000000011111

- **Permutation Encoding** – Useful in ordering problems such as the Traveling Salesman Problem (TSP). Example. In TSP, every chromosome is a string of numbers, each of which represents a city to be visited.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Encoding Methods (contd.)

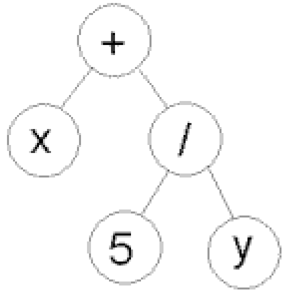
- **Value Encoding** – Used in problems where complicated values, such as real numbers, are used and where binary encoding would not suffice.

  Good for some problems, but *often necessary to develop some specific crossover and mutation techniques for these chromosomes.*

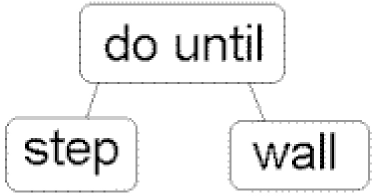
Chromosome A	1.235 5.323 0.454 2.321 2.454
Chromosome B	(left), (back), (left), (right), (forward)

Encoding Methods (contd.)

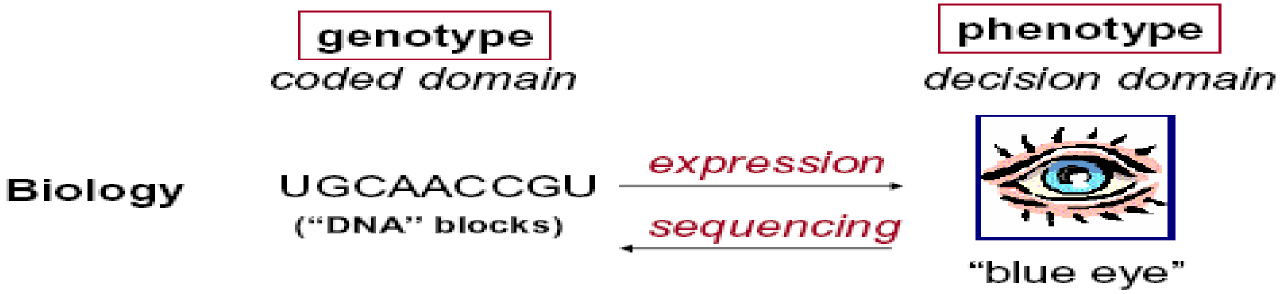
- **Tree Encoding –** This encoding is used mainly for evolving programs or expressions, i.e. for Genetic programming.
- **Tree Encoding -** every chromosome is a tree of some objects, such as values/arithmetic operators or commands in a programming language.



(+ x (/ 5 y))



(do_until step wall)



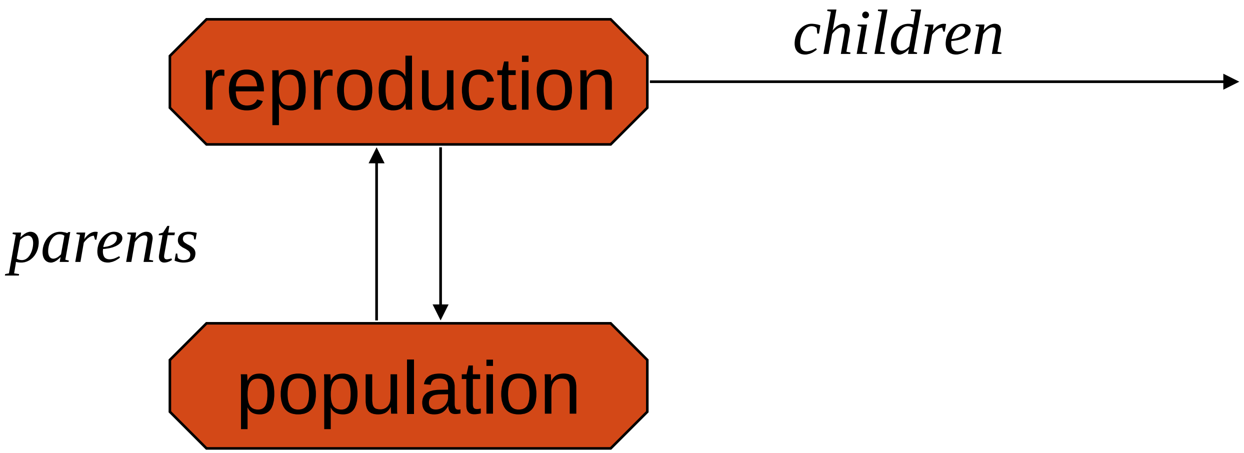
Citation:

http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-888Spring-2004/D66C4396-90C8-49BE-BF4A-4EBE39CEAE6F/0/MSDO_L11_GA.pdf

Initialization

- Start with a population of randomly generated individuals, or use
  - A previously saved population
  - A set of solutions provided by a human expert
  - A set of solutions provided by another heuristic algorithm

Reproduction

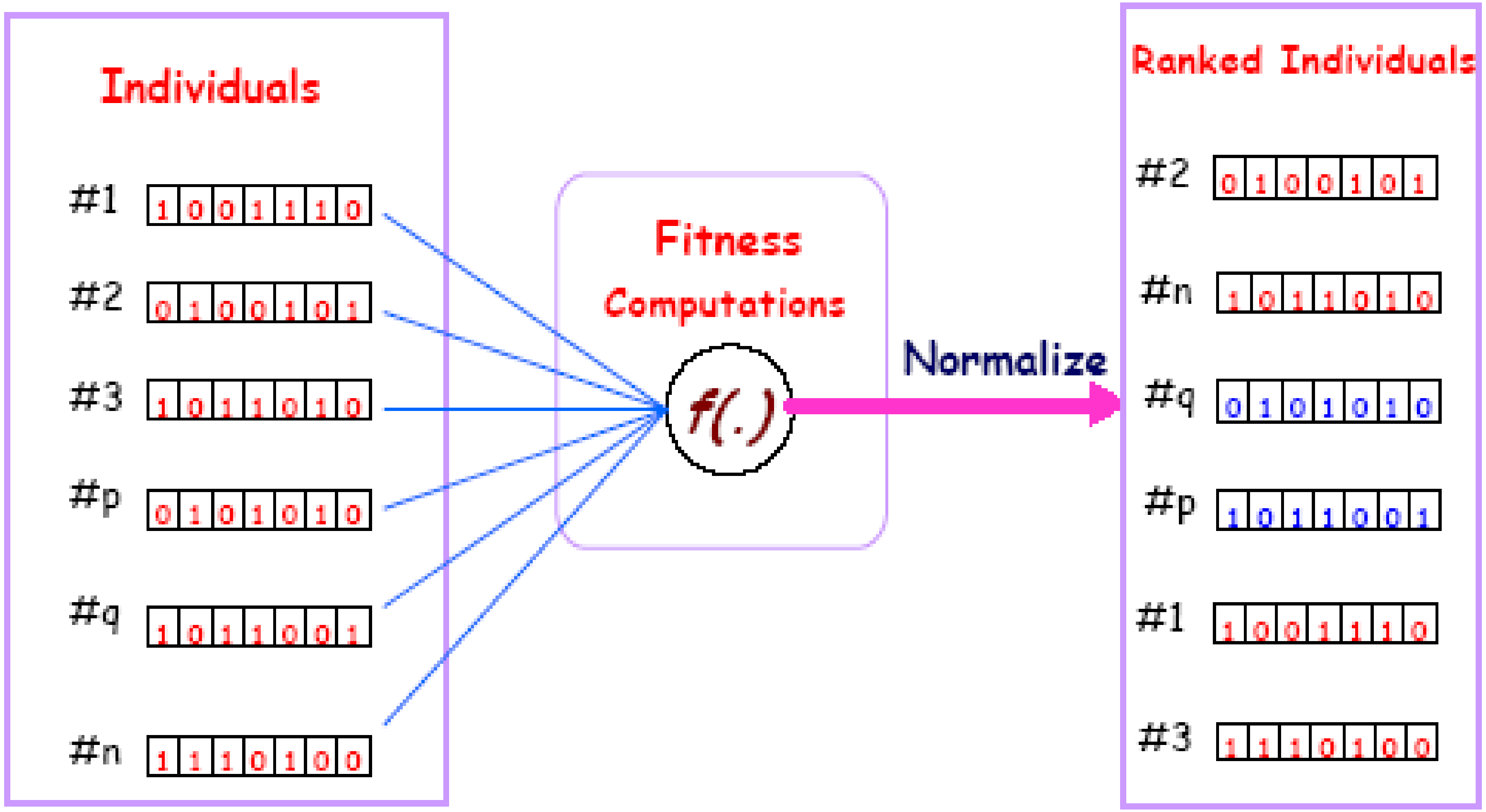


Parents are selected at random with selection chances biased in relation to chromosome evaluations.

Selection -Fitness Function

- 목적 함수(objective function) 즉, 최적화 하고자 하는 함수는 각 개체의 적합도를 평가하는 기반이 된다.
- 목적함수의 값의 범위는 문제마다 다르기 때문에 보통 정해진 구간 사이의 양수값을 갖도록 표준화된 값을 사용한다. 즉, 표준화하기 이전의 적합도의 값을 **raw fitness**라고 하며 표준화되어서 실제로 개체 선택의 기준이 되는 함수를 적합도 함수 (fitness function)라고 한다. 집단 중 다음단계로 교배를 수행하는 개체의 생존 분포를 결정하는 것을 의미한다.
- **raw fitness**를 표준화(또는 스케일링)하는 방법
  - 선형 표준화(**linear scaling**)
  - σ절단(σ **truncation**) :계산된 적합도의 표준편차를 고려하는 방법
  - 거듭제곱 표준화(**power law scaling**)

A Fitness Function



Type of Selection

- 자연선택(natural selection)현상을 모델링하여 잘 적응한 해들은 살아남고 잘 적응하지 못한 해들은 도태되도록 유도한다.
- 방법
  - 기본 모델 : 적합도 비례 선택(proportionate selection), 룰렛 선택법(roulette selection)
    - 각 개체 si 의 적합도 f(si)(＞0), i = 1, …, N 의 총합을 구해, 각 개체 si의 선택 확률을 다음과 같이 정하는 방법

$$p_i = \frac{f(s_i)}{\sum_{j=1}^{N} f(s_j)}$$

  - 엘리트 보존 선택(elitist preserving selection)
    - 확률에 따라 개체를 선택하여 교배 및 돌연변이의 결과로 특별히 좋은 해가 소실되는 것을 막기 위해 가장 좋은 해를 보존하여 다음 세대에 남기는 방법
    - 일반적으로 다른 선택 방법과 융합하여 사용가능

Type of Selection(cont.)

- 기대치 선택법(expected-value selection)
  - 적합도 비례선택의 문제점은 개체군의 크기가 크지 않을 경우에 적합도가 정확히 반영되지 않을 가능성이 있다. 이런 문제점의 대안으로 제안된 것이 기대치 선택법이다. 적합도에 대한 각 개체의 확률적인 재생 개체수를 구하여 선택하는 것이 기본적인 방법이다
- 순위 선택법(ranking selection)
  - 적합도의 크기 순서에 따라 순위를 매긴 후 순위에 따라 다음세대에 자손을 남길 확률을 결정하는 방법
- 토너먼트 선택법(tournament selection)
  - 개체군 중에서 일정한 개수의 개체를 임의로 선택하여 그 중에 최고의 적합도를 가지는 개체를 다음 세대에 남기는 방법(토너먼트 방식).
  - 다음 세대의 개체수가 모두 찰 때까지 반복적으로 계속 수행
- 기타 GENITOR 알고리즘 등

GA Operators(cont.)

③ 균일 교배(uniform crossover): 마스크에 의한 교배

$$p_1 = a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10} \qquad s_1 = a_1\, b_2b_3a_4a_5a_6\, b_7a_8\, b_9b_{10}$$

$$m = 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0 \quad \rightarrow \quad m = 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0$$

$$p_2 = b_1b_2b_3b_4b_5b_6b_7b_8b_9b_{10} \qquad s_2 = b_1\, \underline{a_2a_3}b_4b_5b_6\, \underline{a_7}b_8\, \underline{a_9a_{10}}$$

④ 부분 일치 교배(partially matched crossover : PMX)

⑤ 순서 교배(ordered crossover : OX)

⑥ 주기 교배(cycle crossover : CX)

$$C = \underline{9}\ 8\ 2\ \underline{1}\ 7\ \underline{4}\ 5\ 10\ \underline{6}\ 3$$

$$D = \underline{1}\ 2\ 3\ \underline{4}\ 5\ \underline{6}\ 7\ 8\ \underline{9}\ 10$$

→ C' = 9 - - 1 - 4 - - 6 -

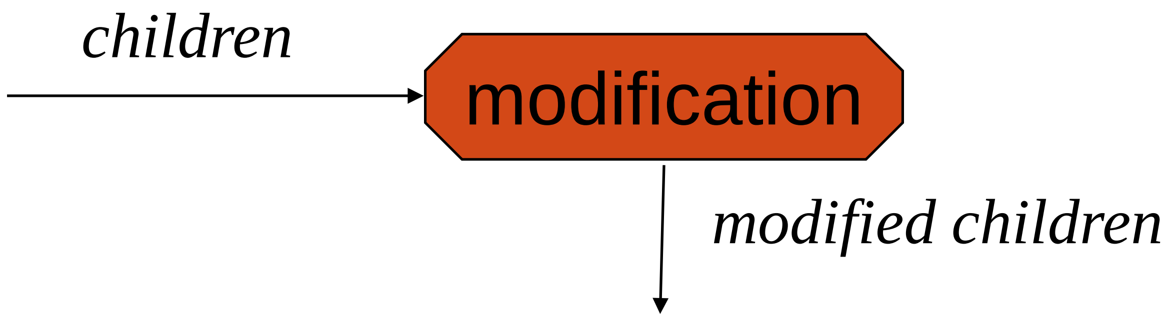
→ C' = 9 2 3 1 5 4 7 8 6 10

D' = 1 8 2 4 7 6 5 10 9 3

Genetic Operators(example)

	Initial strings	*Crossover Mask*	*Offspring*
Single-point crossover:	11101001000	11111000000	11101010101
	00001010101		00001001000
Two-point crossover:	11101001000	00111110000	11001011000
	00001010101		00101000101
Uniform crossover:	11101001000	10011010011	10001000100
	00001010101		01101011001
Point mutation:	11101001000		11101011000

Chromosome Modification



- Modifications are stochastically triggered
- Operator types are:
  - Mutation
  - Crossover (recombination)

GA Operator: Mutation

- 개체의 각 유전자자리의 유전자에 대하여 일정한 돌연변이 확률(pm) 을 적용하여 대립 유전자의 값으로 바꾸는 것
- 개체에 근접한 새로운 개체를 생성하는 국소적인 랜덤 탐색의 일종
- 또한 집단에서 잃어버린 유전형질을 복구하여 다양성을 유지하기 위한 수단으로도 사용됨. 이때 전형적인 돌연변이 확률은 0.05이하
- (예) 세 번째 비트인 a3 가 A3 로 바뀌었음

$$p = a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} \quad \rightarrow \quad s = a_1 a_2 \underline{A_3} a_4 a_5 a_6 a_7 a_8 a_9 a_{10}$$

$\uparrow$

Other GA Operators

- 치환(displacement): 염색체의 일부분을 다른 염색체의 일부분으로 대치하는 방법
- 역위(inversion) : 두 개의 역위점을 구하여 그 사이의 비트들의 순서를 반대로 바꾼다

$$p = a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10} \quad \rightarrow \quad s = a_1a_2a_3\underline{a_8a_7a_6a_5a_4}a_9a_{10}$$

- 중복(duplication) :염색체상의 코드의 일부분을 중복시킨다.
- 추가(addition) :염색체상에 어떤 길이의 부분 문자열을 삽입시킨다. 이 결과 염색체의 길이가 길어지게 된다.
- 제거(deletion) : 염색체상에 어떤 길이의 부분 문자열을 제거시킨다. 이 결과 염색체의 길이가 짧아지게 된다.

Mutation: Local Modification

Before:	(1 0 1 $\boxed{1}$ 0 1 1 0)
After:	(0 1 1 $\boxed{0}$ 0 1 1 0)

Before:	(1.38 $\boxed{-69.4}$ 326.44 0.1)
After:	(1.38 $\boxed{-67.5}$ 326.44 0.1)

- Causes movement in the search space (local or global)
- Restores lost information to the population

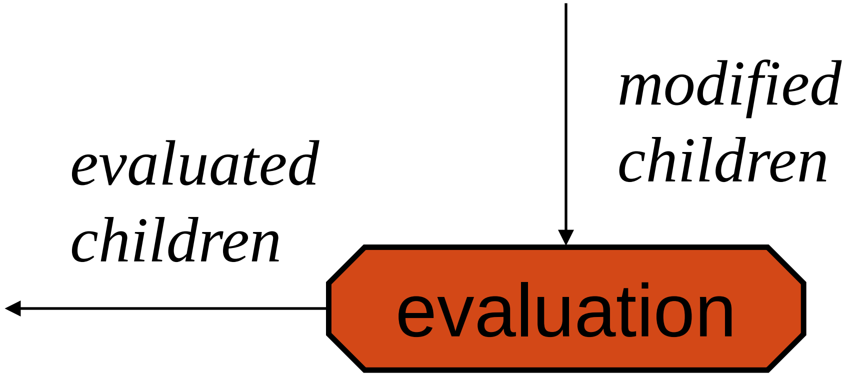
Crossover: Recombination

$$
\begin{array}{lcccl}
 & \overset{*}{} & & & \\
\text{P1} & (0\ 1\ \boxed{1}\ 0\ 1\ 0\ 0\ 0) & \longrightarrow & (0\ 1\ \boxed{0}\ 0\ 1\ 0\ 0\ 0) & \text{C1} \\
\text{P2} & (1\ 1\ \boxed{0}\ 1\ 1\ 0\ 1\ 0) & & (1\ 1\ \boxed{1}\ 1\ 1\ 0\ 1\ 0) & \text{C2}
\end{array}
$$

Crossover is a critical feature of genetic algorithms:

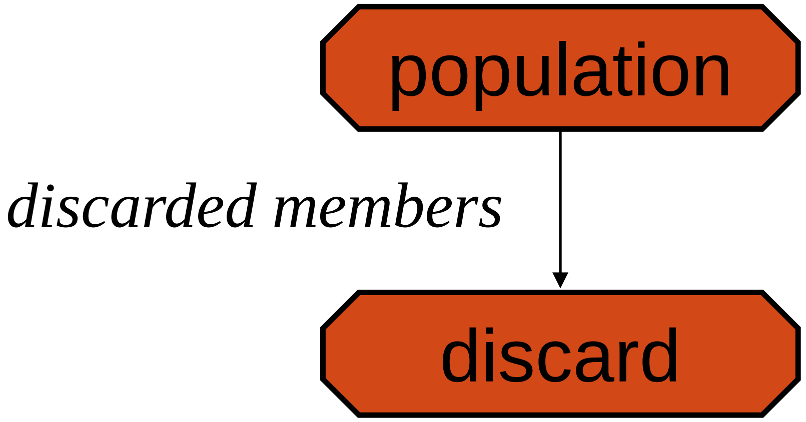
- It greatly accelerates search early in evolution of a population
- It leads to effective combination of schemata (subsolutions on different chromosomes)

Evaluation



- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving

Deletion



- *Generational* GA:
  entire populations replaced with each iteration
- *Steady-state* GA:
  a few members replaced each generation

A Simple Example

The Traveling Salesman Problem:

Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

Representation

Representation is an ordered list of city numbers known as an *order-based* GA.

1) London	3) Dunedin	5) Beijing	7) Tokyo
2) Venice	4) Singapore	6) Phoenix	8) Victoria

CityList1 (3 5 7 2 1 6 4 8)

CityList2 (2 5 7 6 8 1 3 4)

Crossover

Crossover combines inversion and recombination:

	*	
Parent1	(3 5 $\boxed{7\ 2\ 1\ 6}$ 4 8)
Parent2	(2 5 7 6 8 1 3 4)
Child	(5 8 $\boxed{7\ 2\ 1\ 6}$ 3 4)

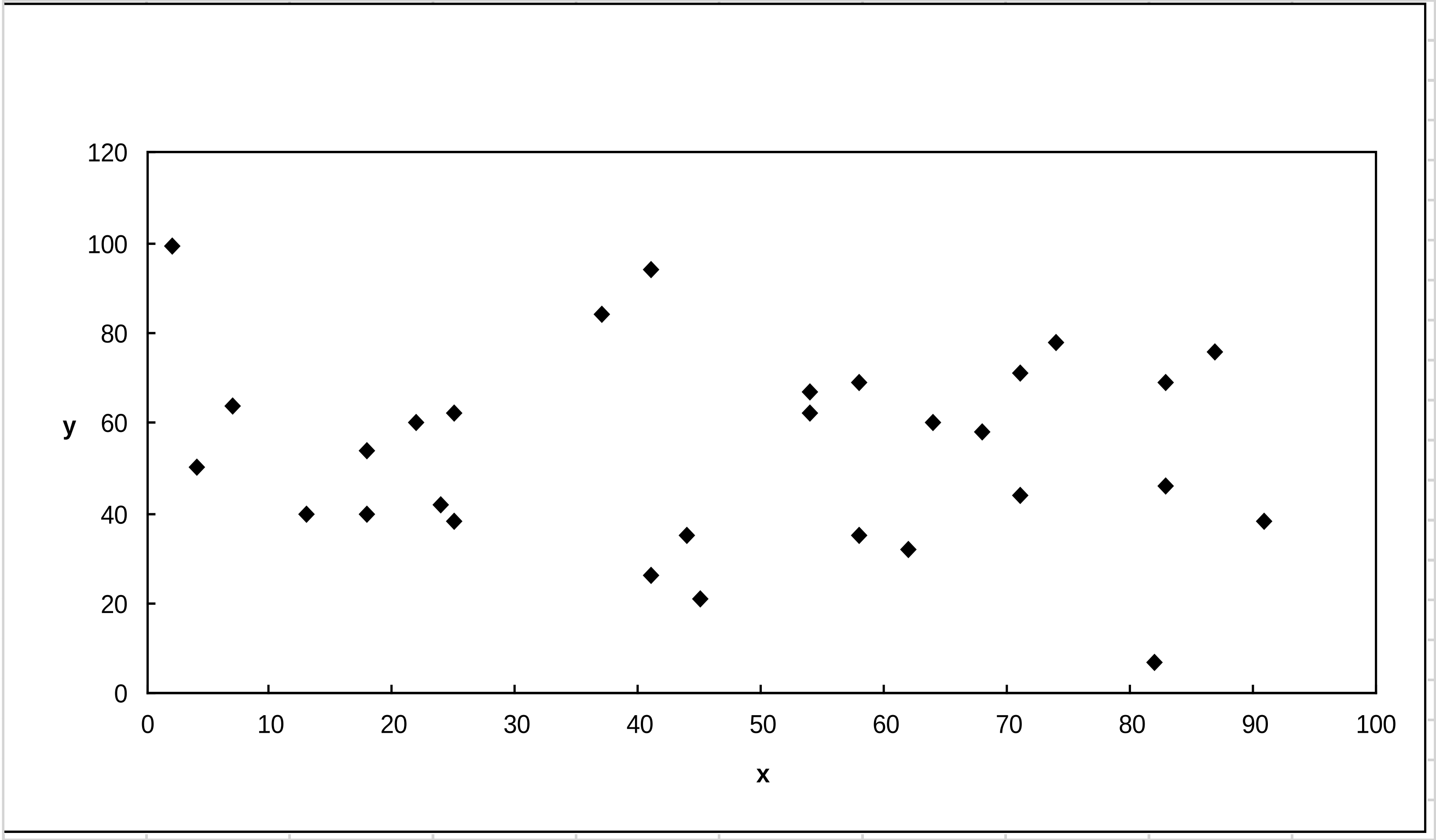
This operator is called the *Order1* crossover.

Mutation

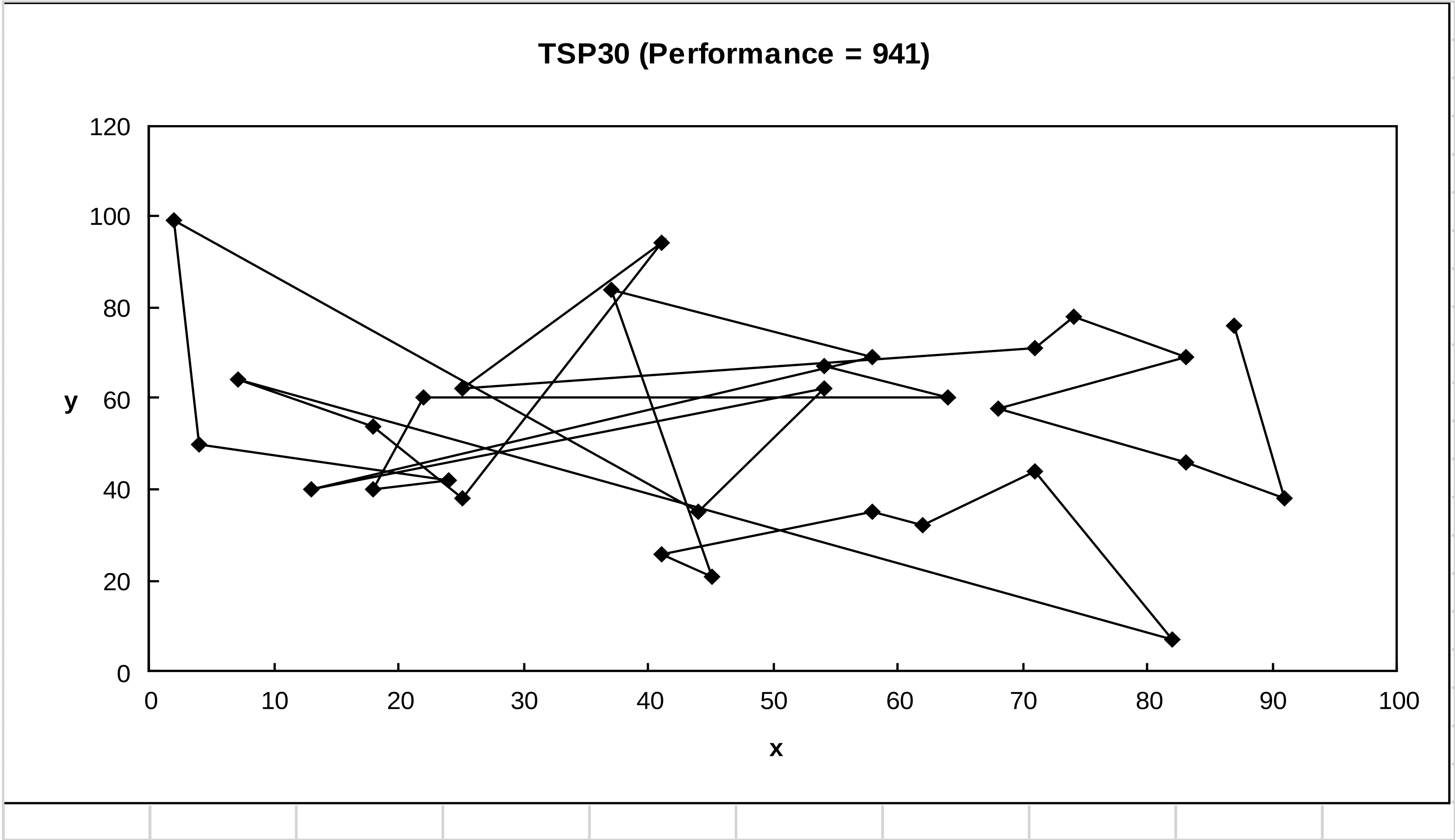
Mutation involves reordering of the list:

			*			*		
Before:	(5	8	$\boxed{7}$	2	1	$\boxed{6}$	3	4)
After:	(5	8	$\boxed{6}$	2	1	$\boxed{7}$	3	4)

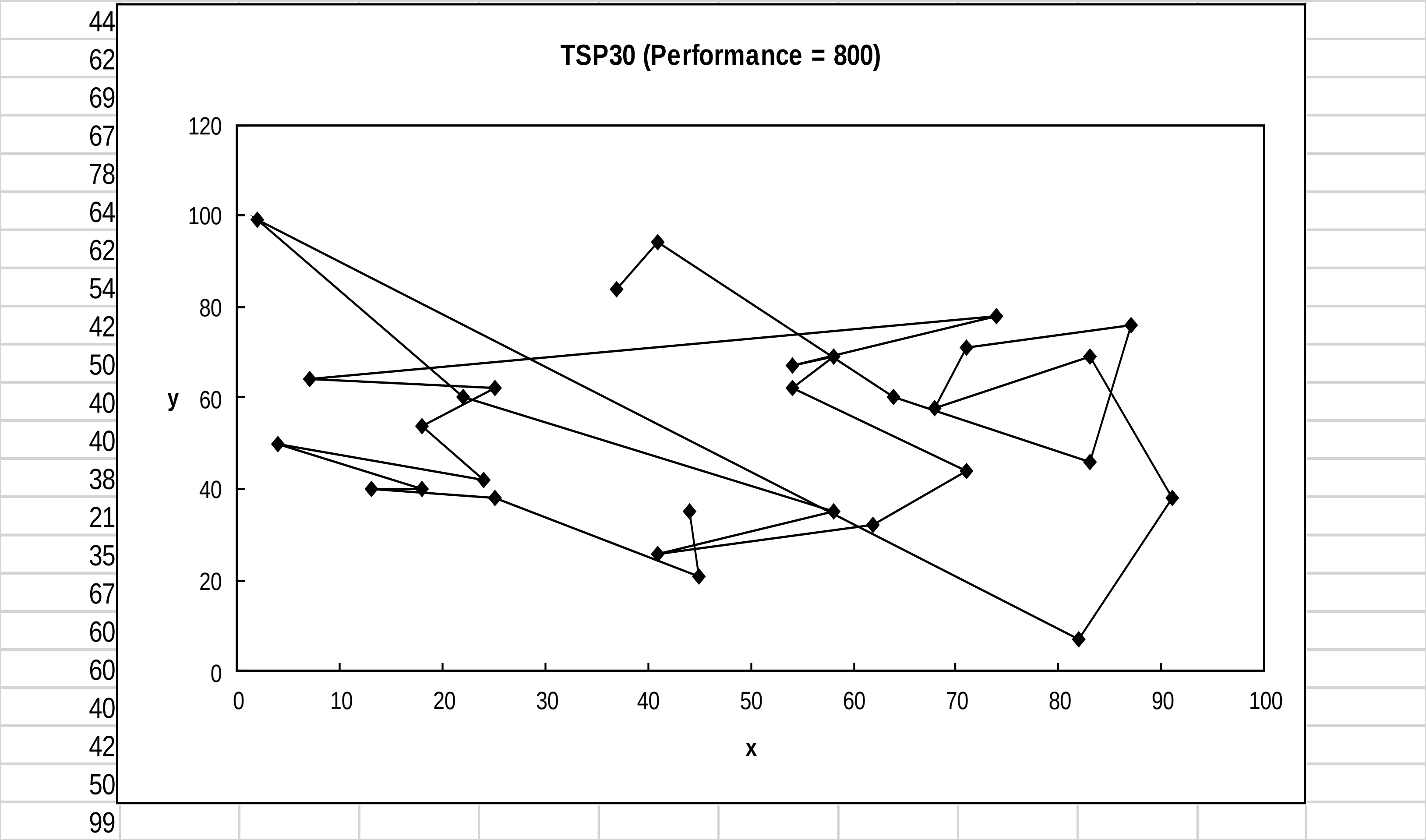
TSP Example: 30 Cities



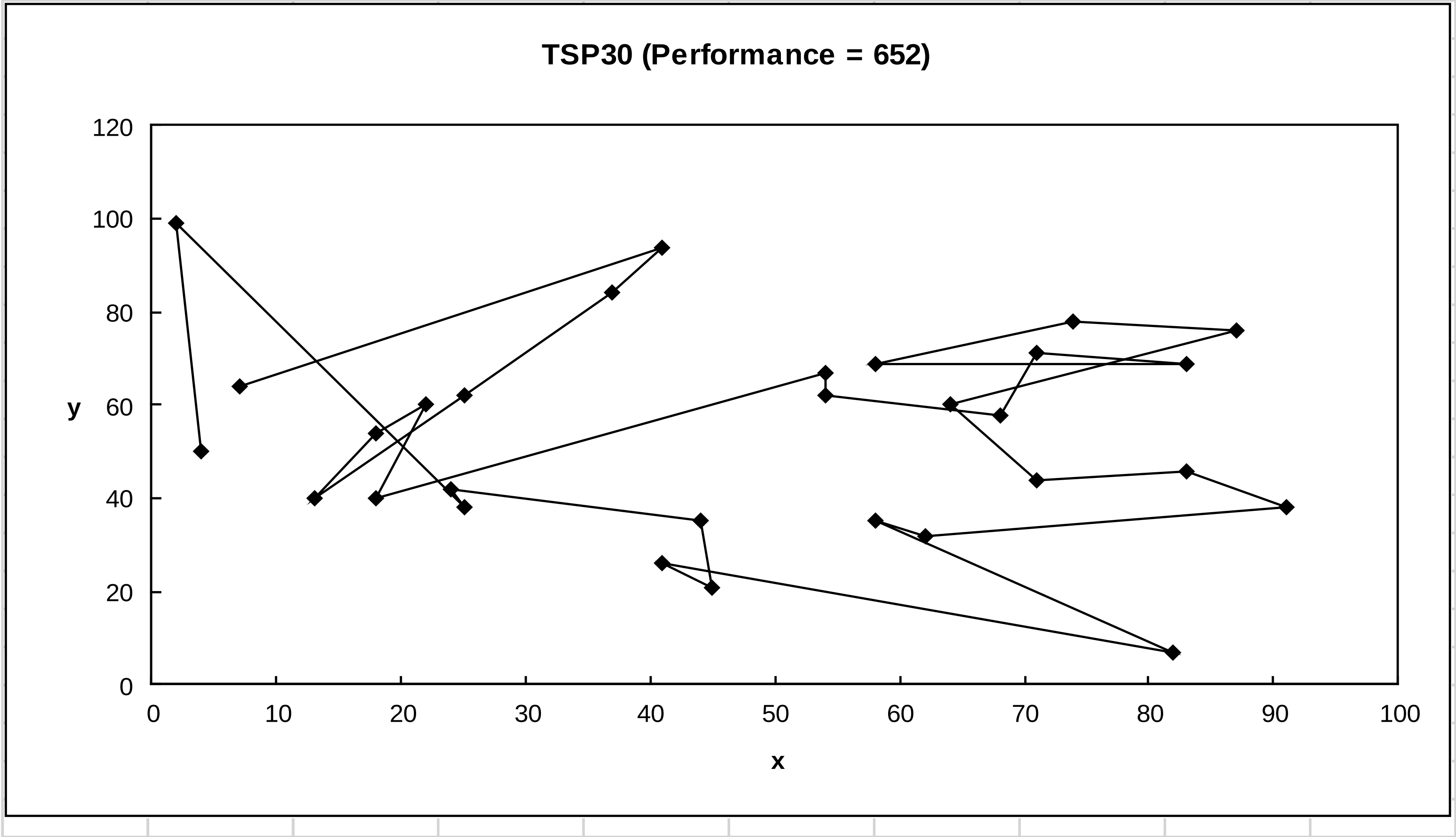
Solution ¡ (Distance = 941)



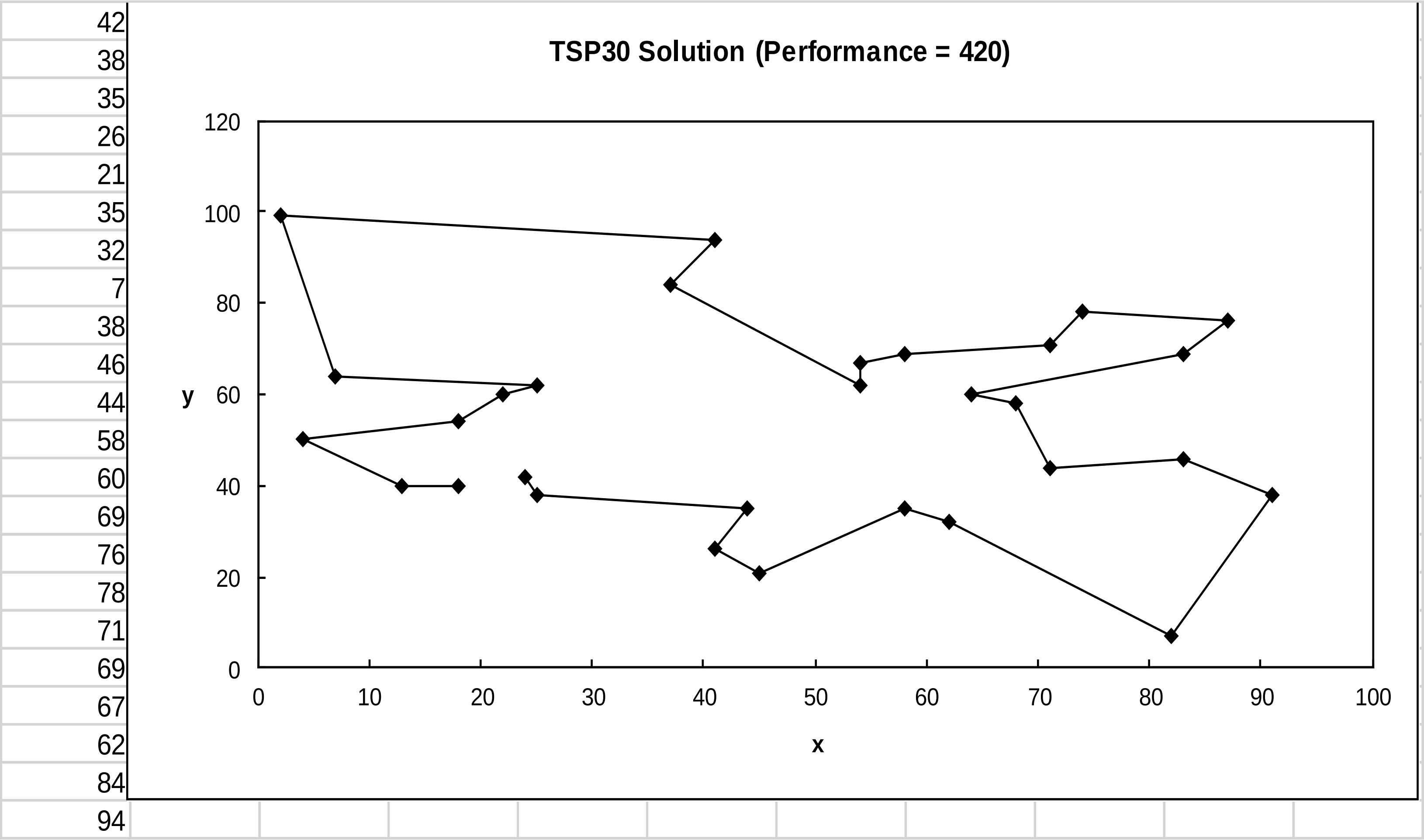
Solution ¡(Distance = 800)



Solution $_k$(Distance = 652)



Best Solution (Distance = 420)



Some GA Application Types

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner뭘 dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning