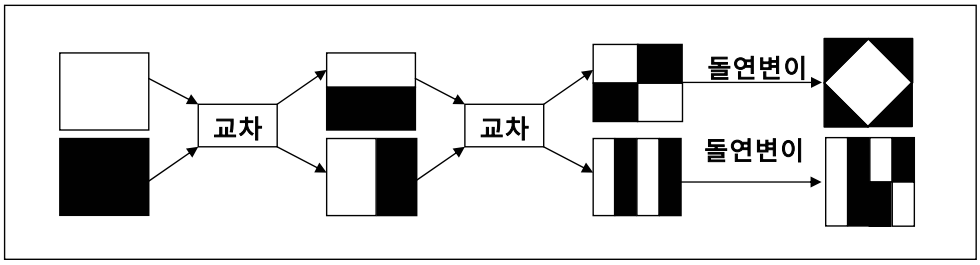
<고급 소프트웨어 실습 II -3주차>

Genetic Algorithm

Genetic Algorithm

- Genetic Algorithm(이하 GA)은 생물의 진화과정으로부터 착안한 컴퓨터를 이용한 문제해결 모델이다. 즉, GA는 주어진 문제에 대한 가능해들을 컴퓨터상의 유전자 형태로 표현한 후 이들을 유전법칙과 적자생존의 법칙에 따라 점차 진화시켜 종국에는 최적해에 이르도록 만드는 방법이다.

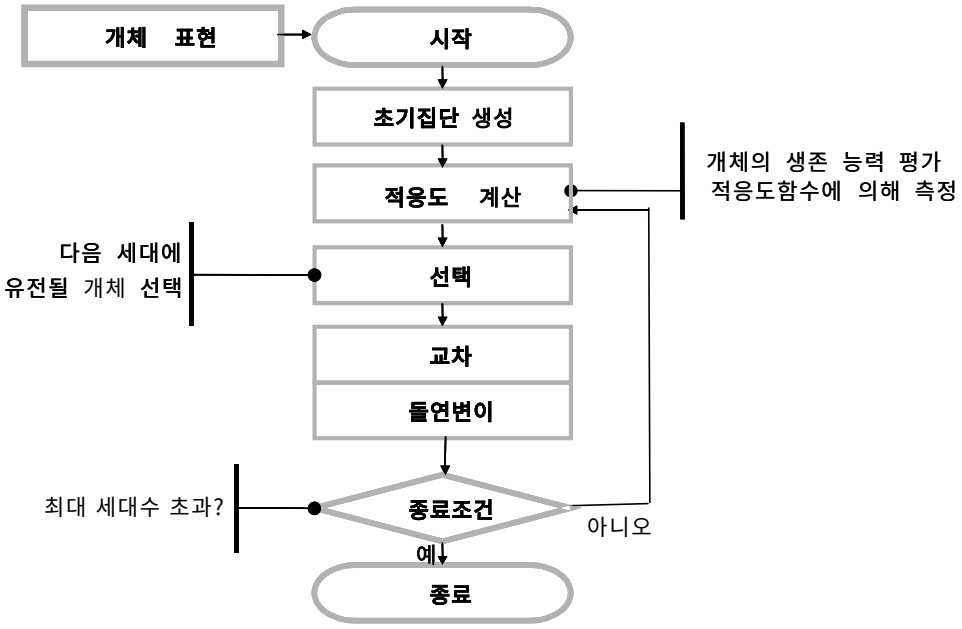


- GA는 아래에 보인 여러 다양한 응용분야에서 사용된다.
  - 조합최적화문제의 풀이
  - 인공생명
  - 전문가 시스템
  - 신경망 이론

GA의 기본 요소들

- 염색체(chromosome) : 정보를 표현하는 자료구조.
  일반적으로는 가능해를 나타내는 일차원 배열
- 유전자(gene) : 염색체가 표현하는 정보.
  즉, 일반적으로는 일차원 배열에 들어가는 데이터
- 개체군(population) : GA의 각 단계가 실행되는 염색체들의 집합.
- 세대(generation) : 개체군을 진화의 순서에 따라 구분한 것
- 적응도(fitness) : 주어진 염색체가 문제의 해에 근접한 정도

GA의 문제해결과정



GA의 문제해결과정

개체표현 : 주어진 문제를 GA에 대해 코드화

자료구조결정 : 가능해를 염색체로 표현하는 방식 설계

적응도함수 설정 : 주어진 염색체의 적응도를 계산하는 방법 결정

초기집단의 생성

n개의 염색체를 임의로 생성하여 초기 개체군, 즉 1세대 생성

종료조건이 만족될 때까지 다음을 반복

적응도 계산 : 적응도 함수에 의해 각 염색체의 적응도를 계산

선택 : 교차에 참여할 염색체들을 선택

교차 : 선택된 염색체들의 유전자를 교환하여 새로운 개체군을 만듦

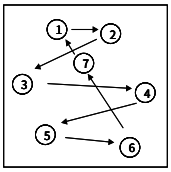
돌연변이 : 교차를 통해 만들어진 새로운 개체군의 유전자를 임의 변화

종료조건확인 : 최대 세대수 초과 여부

GA의 문제해결과정

- 개체표현 : 주어진 문제를 GA에 대해 코드화
  - 자료구조결정
    - 문제에 대한 해(solution)를 일차원 배열로 표현

예) TSP의 여행경로를 염색체로 표현



1	2	3	4	5	6	7

  - 적응도함수 설정

적응도 함수는 각 염색체의 적응도 함수를 계산해주는 방법을 의미한다.

- Traveling Salesperson Problem의 경우
  TSP는 문제해결을 위해 전체여행거리를 최소화 시켜야 하므로 적응도는 전체여행거리가 되며, 적응도 함수는 이를 계산하는 함수이다.
- Knapsack Problem의 경우
  Knapsack문제는 물건들의 가치총합을 최대화 시키는 문제이므로 적응도는 Knapsack에 담긴 가치의 총합이며 적응도 함수는 이를 계산한다.

GA의 문제해결과정

- 초기집단생성
  - n개의 염색체로 이루어진 개체군, 즉 초기 개체군(1세대)를 생성한다. 이를 위해 다음과 같은 방식들을 취한다.
    - 일반적으로 임의의 유전자 값을 갖도록 한 염색체를 n개 만든다.
    - 다만, 문제의 조건에 따라 유전자에 임의의 값을 넣을 수 없는 경우도 있다. 예컨데 TSP의 경우, 중복되어 방문하는 도시가 발생하지 않도록 해야 하므로 이를 피하기 위해 다음과 같은 방법을 사용할 수 있다.

      i) 이미 유전자에 포함된 값을 기록하여 같은 값을 가지지 못하도록 검사한다.
      ii) 모든 도시를 하나씩 포함하는 염색체를 만든 후, 유전자들의 순서를 임의로 뒤섞는다.

- 적응도 계산
  - 앞서 정의된 적응도 함수로 각각의 염색체가 가지는 적응도를 계산한다. 이때 계산된 적응도는 뒤의 선택과 교차참여율을 결정하는데 이용된다.

GA의 문제해결과정 : 선택

- 선택
  - 미리 정의된 교차율에 따라 개체군에서 일정비율의 염색체를 선택하여, 교차후보군으로 삼는다.
  - 교차율
    전체 개체군 중, 교차에 참여할 염색체의 비율을 말한다.
  - 교차율 m/n
    n개의 염색체에서 교차 후보군으로 염색체 m개를 선택한다.
    (m은 개체수보다 작은 임의의 수).

이 때, 염색체 선택을 위해 주로 다음과 같은 방식을 취한다.

- 일반적으로 적응도가 높은 m개의 염색체를 선택하여 교차에 참여하게 한다.
- 토너먼트전략 : 개체군에서 m보다 작은 수의 염색체들을 임의로 고른 후, 그 중 가장 적응도가 높은 염색체 하나를 선택한다. 이 과정을 m번 반복하여 m개의 개체를 선택한다.
- 엘리트 보존 전략 : 적응도가 높은 염색체 중에 일정한 개수 p의 염색체를 그대로 다음 세대로 넘긴다(p < n).

GA의 문제해결과정 : 교차

□ 교차(crossover)

- 두 염색체를 선택하여 이들의 일부 유전자를 서로 교환하여 새로운 염색체를 생성하는 방법이다.
- 교차 참여율
  - ➢ 하나의 염색체가 교차에 참여하기 위하여 선택될 확률을 의미한다.
  - ➢ 교차 참여율의 계산방법
    - 적응도 비례에 의한 계산

$$\text{교차참여율} = \frac{\text{염색체의 적응도}}{\text{교차 후보군내 염색체의 적응도 총합}}$$

    - 순위에 의한 계산
      적응도에 따라 교차 후보군내의 염색체에 순위를 정하고 이 순위에 따라 미리 정의된 확률값을 교차 참여율로 부여한다.

GA의 문제해결과정 : 교차

- 교차의 종류
  - ➢ 단순교차 : 유전자의 한 지점(교차점)을 기준으로 교차가 일어난다. 일반적으로 교차점은 임의로 선택한다.

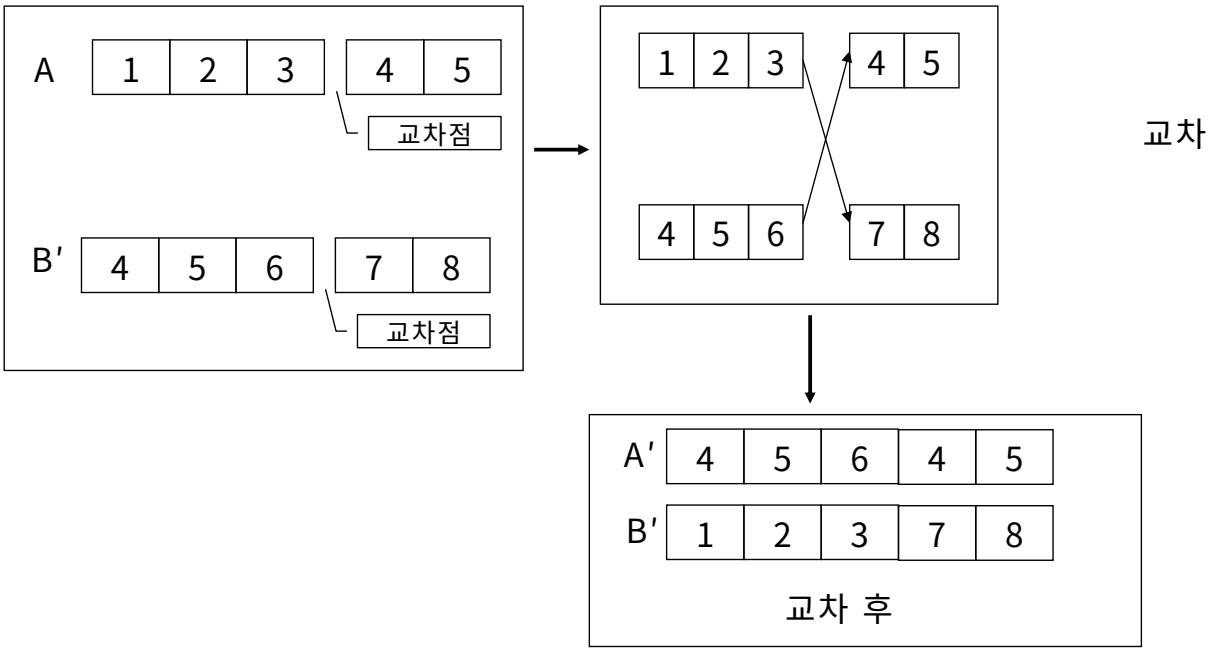
부모세대 자식세대

  - ➢ 복수점교차 : 두 개 이상의 교차점을 기준으로 교차가 일어난다.

부모세대 자식세대

  - ➢ 기타 : 일정교차(uniform crossover), 부분일치교차(Partially Matched Crossover), 순위교차(Ordered Crossover), 주기교차(Cycle Crossover) 등
- m개의 염색체로부터 다음 개체군을 형성할 n개의 개체군을 생성할 때까지 교차를 반복한다. 단, p개의 염색체에 대해 엘리트보존전략을 사용하는 경우 교차를 통해 (n-p)개의 염색체를 생성한다.

GA의 문제해결과정 : 교차 (임의의 염색체)



GA의 문제해결과정 : 돌연변이

- 돌연변이(mutation)

  돌연변이는 특정 염색체 내에서 유전자의 변이를 말한다. 일반적으로 미리 정의된 돌연변이율에 따라 특정 자리를 임의의 값으로 대치하거나, 동일염색체 내의 유전자를 임의로 상호교환하는 방식을 취한다.

  ※ 교차와 돌연변이의 결과로 나타낸 염색체가 무조건 새로운 개체군에 참여하는 것은 아니다. 예를 들자면, TSP의 경우는 여행경로상에 도시가 중복되어 나타나면 안되며 Knapsack의 경우에는 배낭에 담긴 무게의 총량이 배낭의 용량을 넘어서는 안된다. 이러한 제약조건을 위배하는 유전자가 있을 경우 염색체를 사용불가능하게 만들므로 이를 치사유전자라 한다. 이를 피하기 위해 교차나 돌연변이에서는 다음과 같은 방식을 취한다.

  - 새로운 염색체가 제약조건을 위배하지 않을 수 있는 방식을 택한다.
  - 실행결과로 제약조건을 위배하지 않는 유전자가 생성될 때까지 반복한다.

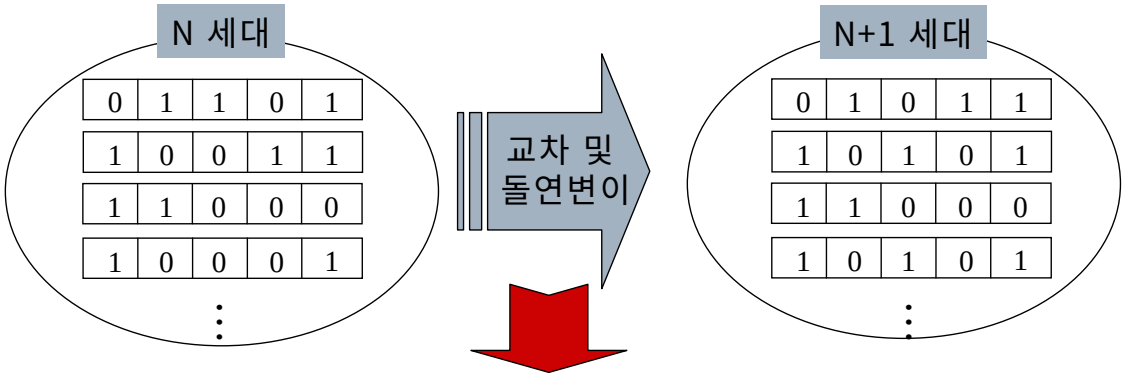
  이중 전자의 방식에 대해 뒤에서 TSP를 다루며 설명할 것이다.

GA의 문제해결과정의 도해(Knapsack의 경우)

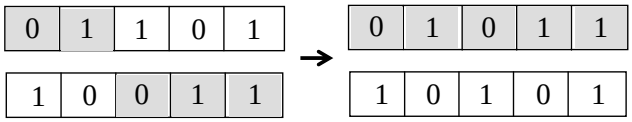
• 개체 (해의 표현) :

x_1	x_2	x_3	x_4	x_5

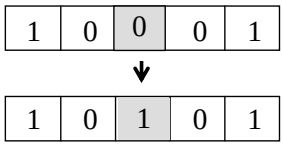
x_i = 1 : i 번째 물건을 담는다.
0 : i 번째 물건을 담지 않는다.



✓ 교차(crossover)



✓ 돌연변이(mutation)



GA의 문제해결과정 : 종료조건

□ 종료조건확인

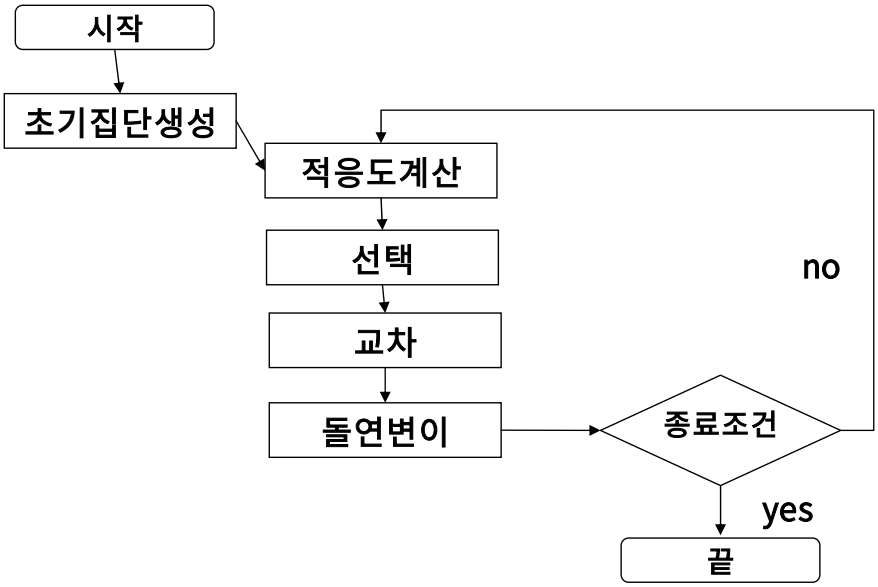
- 개체군의 세대가 미리 정의된 최대 세대수에 이르렀는지를 확인한다.
  - ➢ 최대 세대수라면 GA의 실행을 종료한다. 단 이 때의 개체군 중에 가장 적응도가 높은 염색체가 최적해에 가장 근접한 해일 수는 있지만, 최적해인지 알 수는 없다.
  - ➢ 최대 세대수를 초과하지 않았다면, 새로 생성된 개체군을 부모세대로 삼아 적응도 계산에서부터 다시 시작한다.

※ GA는 교차와 돌연변이를 통해 세대를 거듭하면서 좀 더 적응도가 높은 염색체를 남긴다. 하지만 최적해에 이르기 위한 세대수를 미리 알 수 있는 방법이 없으며, 어떤 염색체가 최적해를 표현하는지를 확인할 수도 없다. 그러므로 GA는 최대 세대수를 정하여 그만큼 교차와 돌연변이를 반복하고, 그 때의 결과만을 보여줄 수 있을 뿐이다.

※ 또한 개체군의 모든 염색체가 일정한 모양으로 수렴한다면, 교차는 의미가 없어진다. 그러므로 개체군의 염색체들이 특정모양으로 수렴하는지의 여부로 종료조건을 삼기도 하는데, 여기서는 이에 대해 고려하지 않는다.

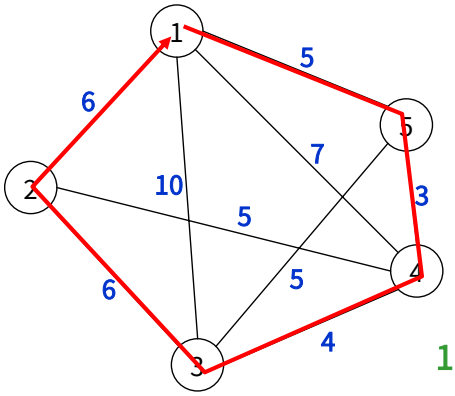
※ 지금까지 GA에서 사용된 개체군의 수(n), 교차율(m / n), 엘리트의 수(p), 최대 세대수, 돌연변이율 등의 수치들은 사용자의 직관에 의해 결정된다. 이러한 조건들의 적당한 값을 구하는 연구가 진행되고 있기는 하나 이것 또한 본 과정의 범위를 넘어선다.

GA의 문제해결과정 : 정리



Traveling Salesperson Problem

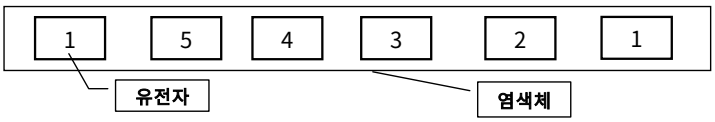
- Traveling Salesperson Problem
  - 널리 알려진 TSP(Traveling Salesperson Problem)은 n개의 도시가 주어지고, 외판원이 각 도시 사이를 여행하는 비용이 주어졌을 때, 가장 적은 비용으로 중복되지 않게 도시를 여행하고 출발한 도시로 돌아오는 문제이다.



1 ⇒ 5 ⇒ 4 ⇒ 3 ⇒ 2 ⇒ 1

Traveling Salesperson Problem with GA

- 개념정의
  - 각 염색체는 모든 도시를 한번씩 순회하는 여행경로를 나타낸다. 즉 각각의 도시가 유전자가 되며, 여행경로는 이 유전자의 배열이다.

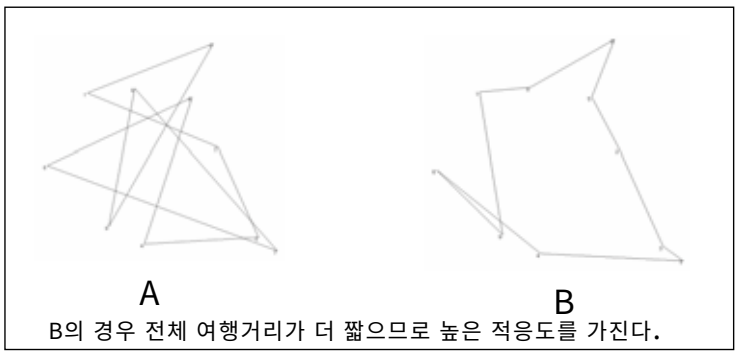


- 적응도 함수 정의
  - 염색체가 나타내는 전체여행경로의 길이

- 초기집단생성
  - 중복되는 도시가 나타나지 않도록 고려하며 임의로 n개의 염색체 생성

Traveling Salesperson Problem with GA

- 적응도 계산
  - 적응도 함수를 바탕으로 각 염색체의 적응도 계산.
    즉, 각 염색체가 나타내는 경로의 전체여행거리 계산



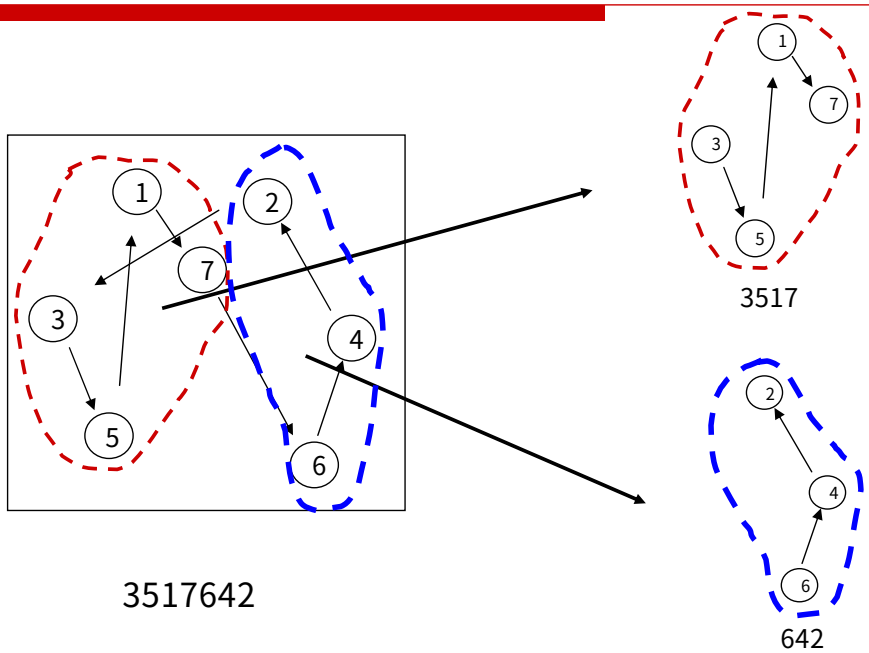
- 선택
  - 개체군에서 m개의 염색체를 적응도 순으로 선택하여 교차에 참여토록함.

Traveling Salesperson Problem with GA

- 교차
  - 교차후보군의 각 염색체에 이의 적응도에 따라 교차참여율 결정
  - 두 개의 염색체를 선택하여 교차
    - 염색체 선택방법
      - 임의의 염색체 c를 후보(P_c = c의 교차 참여율)로 선택한 후 이를 확률 P_c로 실제 선택여부를 결정.
  - 교차 방법
    - 부분 일치 교차 (실습문제에서 예를 들어 설명)
      - 생성된 염색체가 항상 tour를 형성하도록 함.

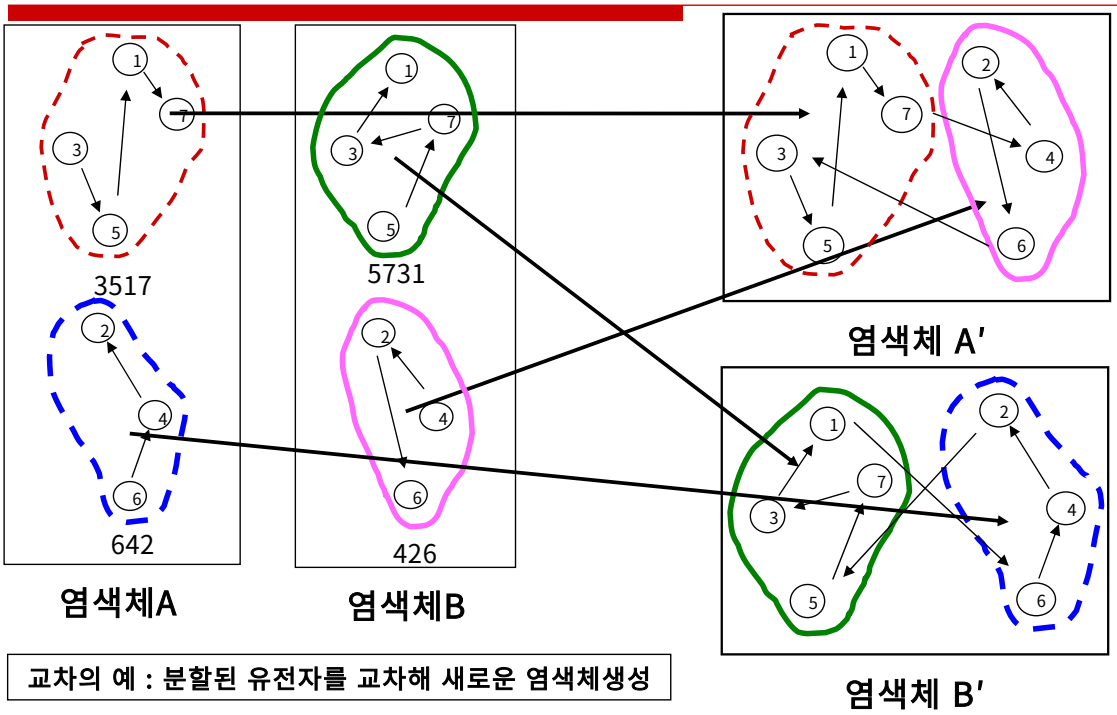
- 돌연변이
  - 교차 결과 나타난 염색체들에 대해 돌연변이율에 따라 염색체의 돌연변이 시킬 것인지, 즉 유전자 값을 임의로 바꿀 것인지를 결정한다.
    - 교차와 마찬가지로 돌연변이 후 생성되는 염색체가 tour를 형성하도록 돌연변이를 수행한다.

Traveling Salesperson Problem with GA



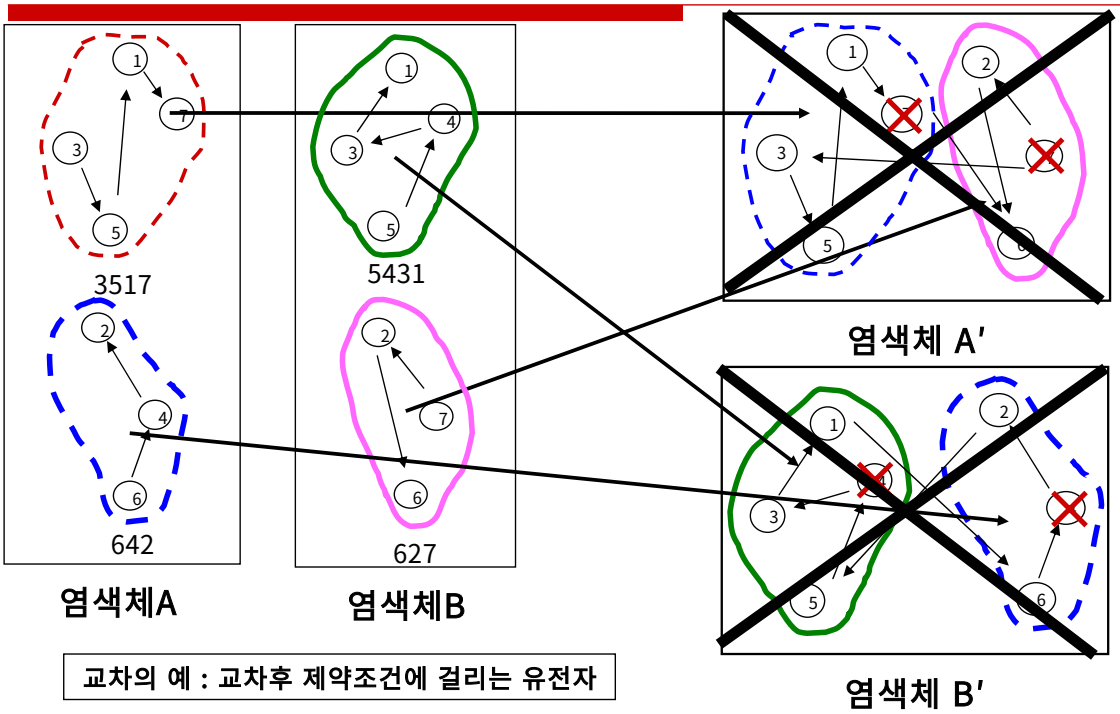
교차의 예 : 교차점을 기준으로 유전자분할

Traveling Salesperson Problem with GA



교차의 예 : 분할된 유전자를 교차해 새로운 염색체생성

Traveling Salesperson Problem with GA

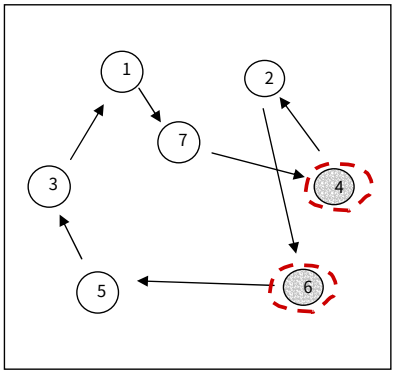


교차의 예 : 교차후 제약조건에 걸리는 유전자

Traveling Salesperson Problem with GA

돌연변이의 예 : 유전자 교환

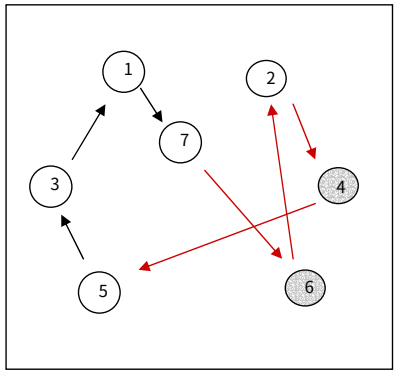
X



3 1 7 **4** 2 **6** 5

4와 6을 교환

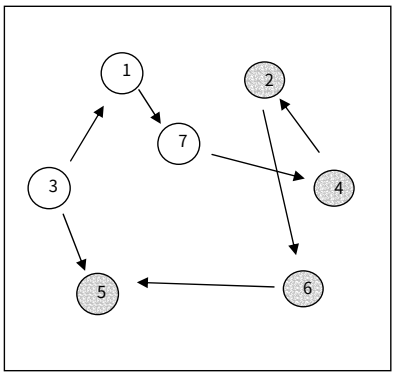
X'



3 1 7 **6** 2 **4** 5

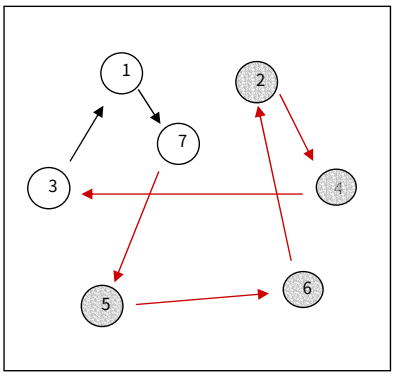
Traveling Salesperson Problem with GA

돌연변이의 예 : 유전자 순서 바꿈



3 1 7 **4 2 6 5**

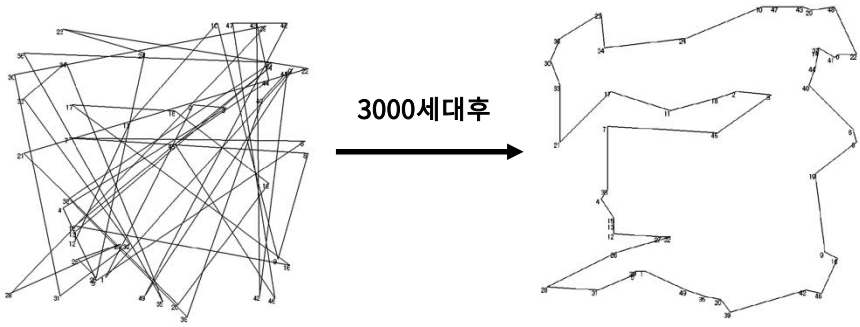
4265의 순서
를 거꾸로



3 1 7 **5 6 2 4**

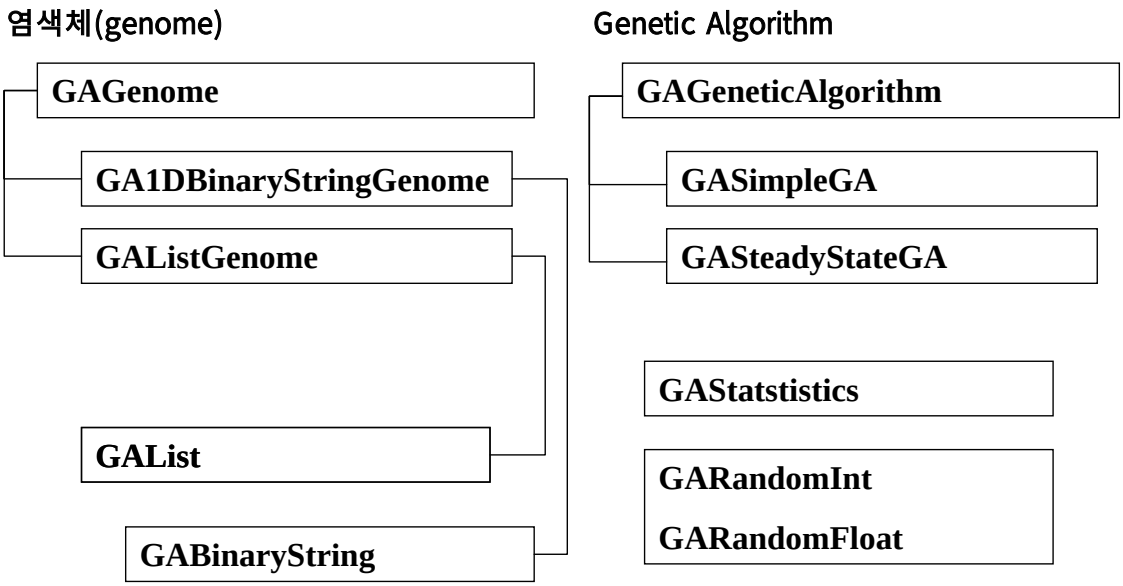
Traveling Salesperson Problem with GA

- TSP에 대한 GA의 복잡도
  - n개의 도시에 대한 TSP의 모든 가능해를 비교하여 그 중 최적해를 찾아낸다고 하면, 가능한 경로가 n!가지가 존재하므로, O(n!)의 복잡도를 갖는다. 하지만, GA의 경우 이보다 더 적은 복잡도로 문제를 해결할 수 있다. 예를 들어 500개의 도시에 대한 TSP문제를 풀기위해, 개체군의 크기를 10, 교차확률을 40%, 교차회수를 10회로 설정한 GA를 적용했을 때, 8084세대 후에 최적해가 나타난다. 다음은 50개의 도시에 대해 200개의 개체군을 가진 GA를 실행한 결과이다.



GALib: A C++ Library of Genetic Algorithm Components

Galib: Class Hierarchy



GALib: A C++ Library of Genetic Algorithm Components

GA1DBinaryStringGenome

: 1과 0으로 이루어진 2진 문자열을 표현하는 염색체형

GAGenome

Method
- ::initializer
- ::mutator
- ::crossover

GABinaryString

Method
- ::gene(int x)
- ::gene(int x, shrot value)
- ::length()
- ::copy

GALib: A C++ Library of Genetic Algorithm Components

GAListGenome

: Linked-list 형태를 표현하는 염색체형

GAGenome

Method
- ::initializer
- ::mutator
- ::crossover

GAList

Method
- ::current()
- ::head()
- ::tail()
- ::next()
- ::prev()
- ::warp()
- ::insert
- ::swap

GALib: A C++ Library of Genetic Algorithm Components

GASimpeGA

: 가장 단순한 형태의 GA

GASteadyStateGA

: 개체군의 일부는 그대로 다음 세대로 전해지도록하는 GA

Common Method
- ::initialize
- ::maximize
- ::minimize
- ::step
- ::done
- ::generation
- ::nGenerations
- ::pMutation
- ::pCrossover
- ::populationSize

GASimpleGA specific Method
없음

GASteadyStateGA specific Method
::pReplacement

GALib: A C++ Library of Genetic Algorithm Components

- float Objective(GAGenome& g)
  - 특정염색체 g가 가지는 값을 계산하는 함수. 적응도함수로 봐도 무방하다.
  - 각 염색체마다 정의되어야하며, GAlib에서 미리 제공하지 않는다.

```
GAListGenome<int> genome(Objective);
/* Objective를 목적함수로 삼고,
   정수를 유전자로 나타내며, 유전자배열을 linked list로 나타낸 염색체 genome을 선언한다. */
float
Objective(GAGenome& g)          // 염색체를 파라메터로 받아온다.
{
  GAListGenome<int> & genome = (GAListGenome<int> &)g;
  float dist = 0;               // 결과값 초기화
  if(genome.head()) {
    for(int i=0; i<ntowns; i++) {
      int xx = *genome.current();        // 현재 도시 번호
      int yy = *genome.next();             // 다음 도시 번호
      dist += DISTANCE[xx][yy];          //  현재 도시에서 다음 도시까지의 거리.
                                         // DISTANCE[x][y] : x에서부터 y까지의 거리를 기록.
    }
  }
  return dist;
}
```

GALib: A C++ Library of Genetic Algorithm Components

- void Initializer(GAGenome& g)
  - 각 염색체가 초기생성될 때 가지게 되는 값을 정해주는 함수
  - Mygenome에 대해, 초기화 함수를 재정의해주기 위해서는
    Mygenome.initializer(::Initializer)

```
void Initializer(GAGenome& g) {
  GAListGenome<int> &child=(GAListGenome<int> &)g;
  while(child.head()) child.destroy(); // destroy any pre-existing list

  child.insert(0,GAListBASE::HEAD); // the head node contains a '0'
  for(int i=1; i<ntowns; i++)
    child.insert(i);
  for(int j=0; j<ntowns; j++)
    child.swap(GARandomInt(0,ntowns-1), GARandomInt(0,ntowns-1));
}
```


GALib: A C++ Library of Genetic Algorithm Components

- int Xover (const GAGenome& g1, const GAGenome& g2, GAGenome* c1, GAGenome* c2)
  - 부모세대의 두 개의 염색체 g1, g2를 교차시켜 염색체 c1, c2를 만든다.

- int Mutator(GAGenome& g, float pmut)
  - 염색체 g에 대해 pmut의 확률로 돌연변이를 일으킨다.

```
int
Mutator(GAGenome& g, float pmut) {
    GAListGenome<int> &genome = (GAListGenome<int> &)g;
    int n = genome.size;   // 리스트의 크기를 구한다


    if (GARandomFloat < pmut) return 0;
    // 돌연변이율: 랜덤값이 pmut보다 작으면 돌연변이를 수행하지 않는다.
    genome.swap(GARandomInt(0, n-1), GARandomInt(0, n-1));
    // 임의의 두자리의 유전자값을 교환한다.
    return 1;     // 돌연변이가 일어났음
}
```

GALib: A C++ Library of Genetic Algorithm Components

□ GAlib의 사용

```
GAListGenome<int> genome(Objective);      // 염색체 정의
genome.initializer(::Initializer);        // 염색체 초기화
genome.mutator(::Mutator);                // 돌연변이 기능 정의
genome.comparator(::Comparator);          // 두개의 염색체 비교기능 정의
genome.crossover(XOVER);                  // 교차 기능 정의


GASteadyStateGA ga(genome);               // 염색체에 대한 GA정의
ga.minimize();                            // objective function을 최소화
ga.pReplacement(1.0);                     // 각 세대마차 교체될 비율(%)
                                          // GASteadyStateGA에서 사용
ga.populationSize(100);                   // 개체군의 크기
ga.nGenerations(1000);                    // 최대 세대수
ga.pMutation(0.1);                        // 돌연변이율
ga.pCrossover(1.0);                       // 교차율


ga.initialize(seed);                      // 초기화시킨다.
while(!ga.done()) {                       // 종료을 충족시킬 때까지 실행
  ga.step();                              // GA의 각 단계 실행
}
genome = ga.statistics().bestIndividual();     // 최적해 출력
```


실습문제 : Traveling Salesperson Problem

※교재에 나온 실습문제를 다음 사항에 유의하여 실습해보도록 한다.

□ 유의사항

- 임의의 도시에서 출발한다.
- 한 도시를 두번 들를 수 없다.
- 처음 도시로 돌아온다.
- 각 도시는 (x,y)형태의 좌표를 가진다.
- 도시간의 여행비용은 거리에 정비례한다.

□ 교차함수작성방법

- 다음은 부분일치교차 방법의 예이다.
  - 두 개의 부모에서 공통되는 서브투어(sub-tour)를 찾는다.
  - 그 서브투어를 하나의 도시로 간주하여 표현을 변경한다.
  - 축약된 염색체로부터 교차를 수행하여 자식을 만든다.
  - 자식을 역변환하여 본래의 염색체 표현으로 돌아간다.

* 서브투어란 두개 이상의 염색체에 공통되어 나타나는 여행경로를 말한다.

실습문제

- 각 도시의 정보를 기록하는데 필요한 자료구조
  - 도시 i의 좌표를 (x_i, y_i)라고 할 때, 각 도시의 좌표를 1차원 행렬 두개를 이용하여 다음과 같이 표현한다.

$$x[i] = x_i,\ y[i] = y_i$$

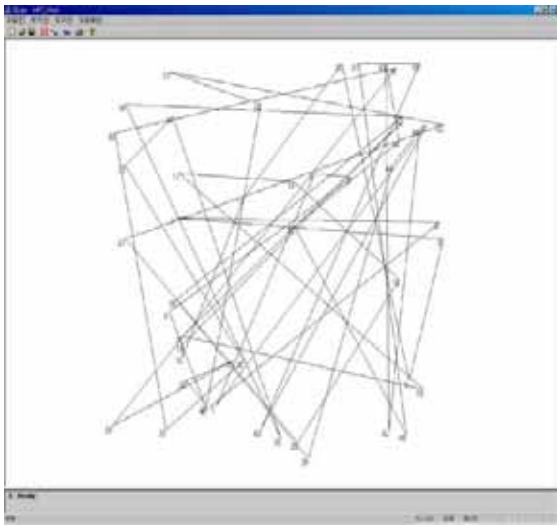
  - 도시 x와 도시y의 거리를 나타내기위해 다음과 같은 2차원 행렬을 사용한다.

DISTANCE[x][y] = x와 y의 거리

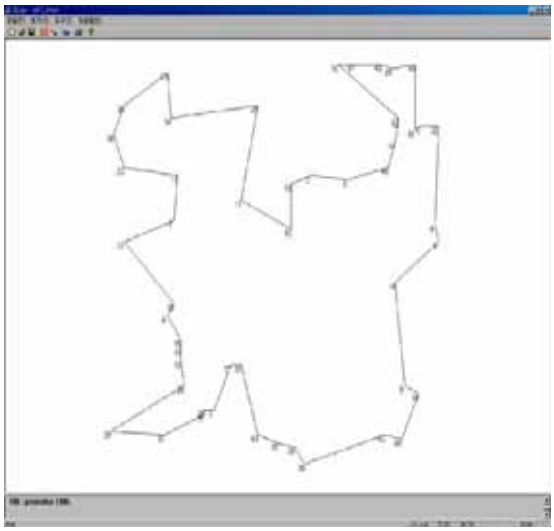
  - 이 때, 두 도시의 거리는 각각의 좌표를 바탕으로 피타고라스 정리를 사용하여 구한다.
  - 각 도시를 노드로 나타내고, 이 노드를 여행순서대로 놓아 리스트를 구성한다.

- 실습방법
  - s19951059.cpp 파일에서 다음 두 함수의 빈 부분을 채워넣는다.
    - int XOVER(const GAGenome& g1, const GAGenome& g2, GAGenome* c1, GAGenome* c2)
    - int Mutator(GAGenome& g, float pmut)
  - XOVER의 경우 , 교차함수 작성방법에서 예시된 것 이외의 방식으로 작성해도 좋다.
- 실험
  - 컴파일과 실행에 성공하면, 주어진 데이터들을 실행시키고, 실행결과를 조교가 확인하도록 한다.
  - 테스트 데이터로 주어진 50.dat에 대해 소스의 go() 함수에서 나타난 GA의 파라메터값, 개체군의 크기, 최대세대수, 교차율, 돌연변이율 등을 다양하게 바꾸어서 실행하고 결과가 어떻게 달라지는지 확인한다.
  - 주어진 소스코드에서 GASteadyStateGA를 GASimpleGA로 대체하여, GA의 실행결과가 어떻게 달라지는지 확인한다.
  - 위 두가지의 테스트 결과를 조교에게 확인받도록 한다.

실습문제의 예



실행 전



실행 후

숙제: 0/1 Knapsack Problem

- 0/1 Knapsack

  0/1 Knapsack Problem은 대표적인 조합최적화문제로서, 무게와 가치를 지닌 n개의 물건들이 있고, 또 m의 무게까지 담을 수 있는 배낭(knapsak)이 있다는 가정에서 시작한다. 물건들의 무게의 총합이 m보다 크므로, 모든 물건을 배낭에 담을 수 없다고 할 때, 물건들의 무게의 총합이 m을 넘지 않는 범위에서 가치의 총합을 최대로 하는 물건의 조합을 구하는 것이 Knapsack Problem의 목적이다. 단, 각 물건은 나눌 수 없고, 배낭에 담거나 말거나의 선택만 가능하다.

- 숙제방법
  - 미리 작성된 화일을 다운 받는다.
  - Objective function을 작성한다.
  - 교차(xover)와 돌연변이(mutator) 함수를 작성한다.
  - 테스트 데이터를 시험해본다.
  - 이전 실습의 Knapsack에서 사용한 데이터를 사용하여 시험해본다.
  - 이전과 같은 결과가 나오지 않는다면, GA에 주어진 파라메터를 수정하여 유사한 결과가 나오도록 조정해본다.
  - 실험결과를 저장한 텍스트 파일과 컴파일된 소스를 압축하여 조교에게 제출하도록 한다.
  - 제출기한은 2003/09/XX까지.

- 유의사항
  - TSP와 달리 중복되는 값에 대한 검사가 필요없으므로 1점 교차만으로도 충분하다.
  - 교차 후 무게의 총합이 m을 초과하면 도태시킨다.