7장 유전 알고리즘: 좋은 것이 남는 진화의 법칙

(Credit: iStock)

출처: http://www.indaily.co.kr/client/news/newsView.asp?nBcate=F1009&nMcate=M1008&nIdx=20506&cpage=1&nType=1

7장 유전 알고리즘: 좋은 것이 남는 진화의 법칙

- **학습 목표**
  - 체험해 봅시다: Ex5_유전 알고리즘.xlsm

    유산을 적절히 잘 분배하기
  - 유전 알고리즘이란?
  - 유전 알고리즘의 구체적 예
  - 유전 알고리즘의 응용

⊙ 체험해 봅시다: 재산을 요령 있게 적절히 분배

- 재산 분배라는 것은 재산이 모두 돈이라면 비례배분으로 간단히 계산이 가능하지만, 일반적으로는 돈 이외의 다양한 형태로 되어 있음.
- 그런 것들은 분배가 불가능하기 때문에 돈과 같은 연속량으로 다루지 못하고 비례배분과 같은 간단한 계산으로는 분배할 수 없음.
- 이것을 컴퓨터로 처리하는 경우에는 **상속인에 대한 분배의 모든 조합**을 생각하고 가장 적정한 분배 비율을 따르게 됨.
- 하지만 유산 물건 수가 적은 경우에는 아무런 문제가 없지만 **물건 수가 많으면 방대한 조합 패턴**이 되어 컴퓨터로 하나하나 조사한다면 방대한 시간이 걸림.
- 그래서 완벽히 유언대로는 아니더라도 **거의 유언에 가까운 분배**를 찾아내기 위해서는 유전 알고리즘이 효과적임.

7.1 유전 알고리즘이란?

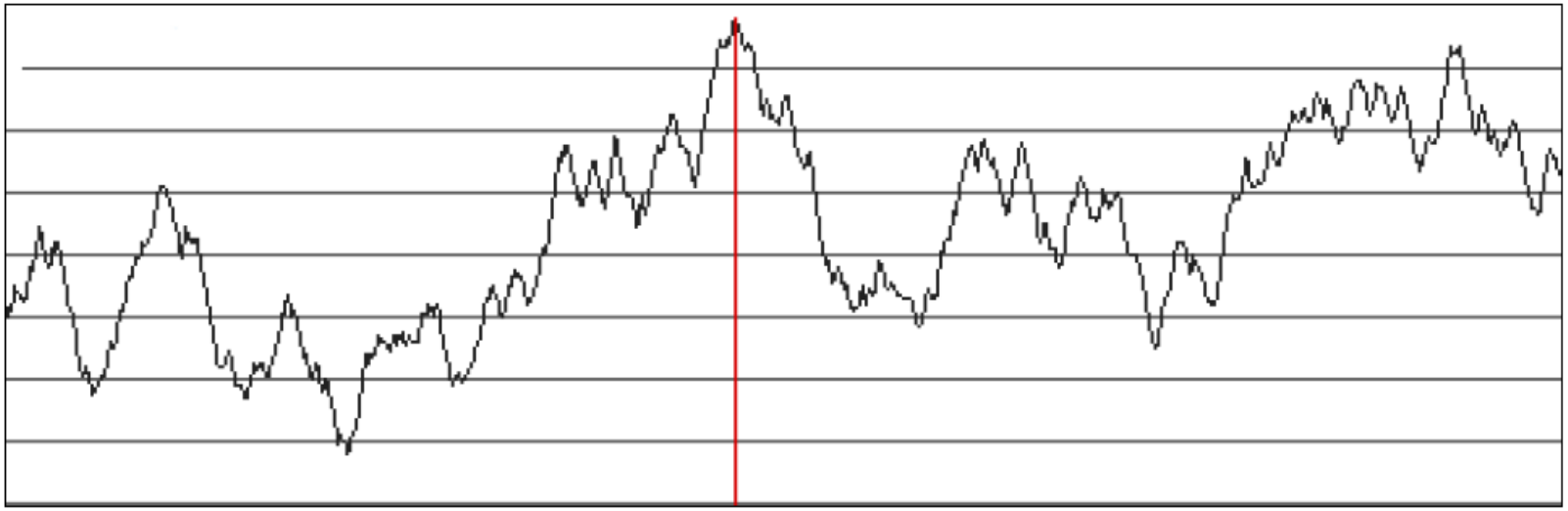
- 유전 알고리즘이 생물의 유전과 진화에서 모방되었다는 것은, 대상 문제의 해를 세대교체를 반복함에 따라 점점 좋은 것으로 변화시켜 나간다는 것을 의미.
- 이 개념은 1960년대부터 있었지만, **홀랜드(John Holland, 1975)**에 의해 개념이 확립되었음.
- 현실에서는 세대교체에 몇 년씩 걸리지만 컴퓨터상에서라면 몇 만 세대의 세대교체를 순식간에 실행할 수 있음.
- 이것으로 문제가 해결된다면 고맙겠지만, 초기에는 이론적인 증명이 부족하였고 그 이후에도 논리적인 검증과 확장 연구가 계속 이루어지고 있음.

7.1 유전 알고리즘이란?

John H. Holland

Born in Indiana 1929.
Author of "Adaptation in Natural and Artificial Systems" written in 1975, providing the basis of genetic algorithms.
Recipient of the McArthur Fellowship.

7.1 유전 알고리즘이란?



7.1.1 유전 알고리즘의 개념

- 우선 대상 문제를 세대 교체할 수 있는 모델로 표현할 필요가 있음. 즉, 대상 문제의 특징을 추출하여 몇 가지 기호열로 표현하고 이 기호열을 세대교체의 대상으로 함.
- 이 기호열이 유전자에 해당하며 이와 같은 모델화가 바로 유전자 설계에 해당함.
- 유전 알고리즘에서는 수식을 사용하지 않는 대신 코딩에 신경씀.
- 유전자 설계 형태는 다양한 기법에 따라 개발되어 있으며, 그 설계 형태에 따라서 유전 알고리즘이 적용 여부가 결정됨.
- 유전자 설계와 동시에 해로서의 가치를 평가하는 지표도 필요하며, 이것을 **적응도**라고 함.

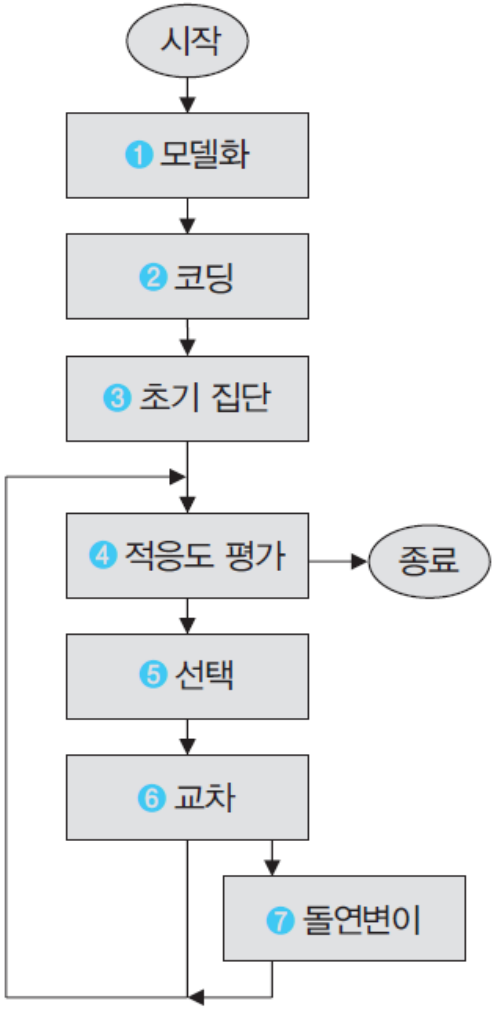
7.1.1 유전 알고리즘의 개념

- 세대교체에는 3가지 실행 단계가 있음.
  - ① **선택**: 다양한 유전자를 갖는 개체 중에서 적응도가 높은 것을 선택.
  - ② **교차**: 그 유전자들 간에 유전자 일부를 교환하여 보다 적응도가 높은 유전자를 갖는개체를 만들어 냄.
  - ③ **돌연변이**: 진화가 정체되지 않도록 가끔 새 유전자를 만들어 끼워 넣음.
- 세부 개념을 정리하면 다음과 같음.
  - **유전자(Gene):** 대상 문제의 특징을 추출하여 세대교체의 대상이 되는 기호열
  - **적응도(Fitness):** 대상 문제에서 요구되는 가치에 얼마나 근접한지를 나타내는 지표
  - **선택(Selection):** 많은 개체들로부터 적응도가 높은 것을 선택해 내는 방법
  - **교차(Crossover):** 개체 간에 유전자를 교환하는 방법
  - **돌연변이(Mutation):** 적응도에 관계없이 임의로 유전자를 끼워 넣는 방법

7.1.2 유전 알고리즘의 처리 순서

- 유전 알고리즘을 적용하는 순서
  1. **모델화:** 대상 문제의 특징을 추출하여 목표 상태를 정의.
  2. **코딩:** 유전자와 적응도 평가 함수를 정의.
  3. **초기 집단:** 적당한 유전자를 갖는 개체를 필요한 개수만큼 생성.
  4. **적응도 평가:** 적응도를 평가. 목표 상태가 되면 종료.
  5. **선택:** 적응도가 높은 유전자를 갖는 개체를 선택.
  6. **교차:** 개체 간에 유전자를 교환.
  7. **돌연변이:** 필요에 따라 적응도에 관계없이 새 유전자를 끼워 넣음.

7.1.2 유전 알고리즘의 처리 순서



❶ 현실 문제로부터 모델을 추출하고, 적응조건, 평가방법을 결정한다.

❷ 유전자형(개체의 표현법), 개체 수, 적응도 평가 함수를 결정한다.

❸ 필요한 수만큼 개체를 생성한다.

❹ 개체수와 집단의 적응도를 평가하여 조건을 벗어나는 것은 배제한다. 최적해가 되는 개체가 있으면 종료. 적응도가 변화하지 않는지 주의가 필요하다.

❺ 집단에서 우수한 개체를 선택한다(선택법).

❻ 두 개체의 유전자를 교환하여 자식 세대의 집단을 생성한다(교차법).

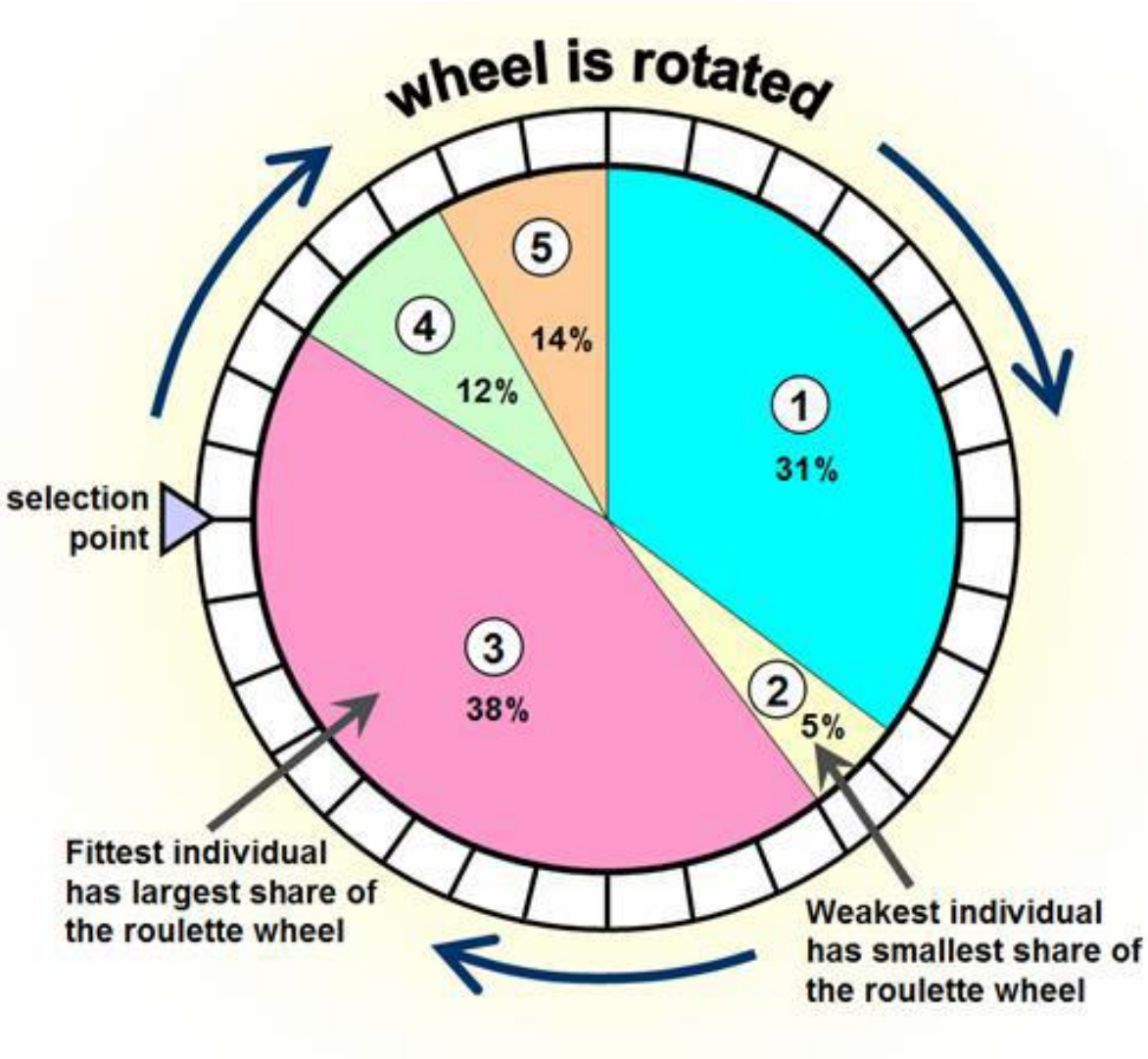
❼ 드물게 교차법과 다른 조합을 삽입한다.

그림 4-1 유전 알고리즘의 처리 순서

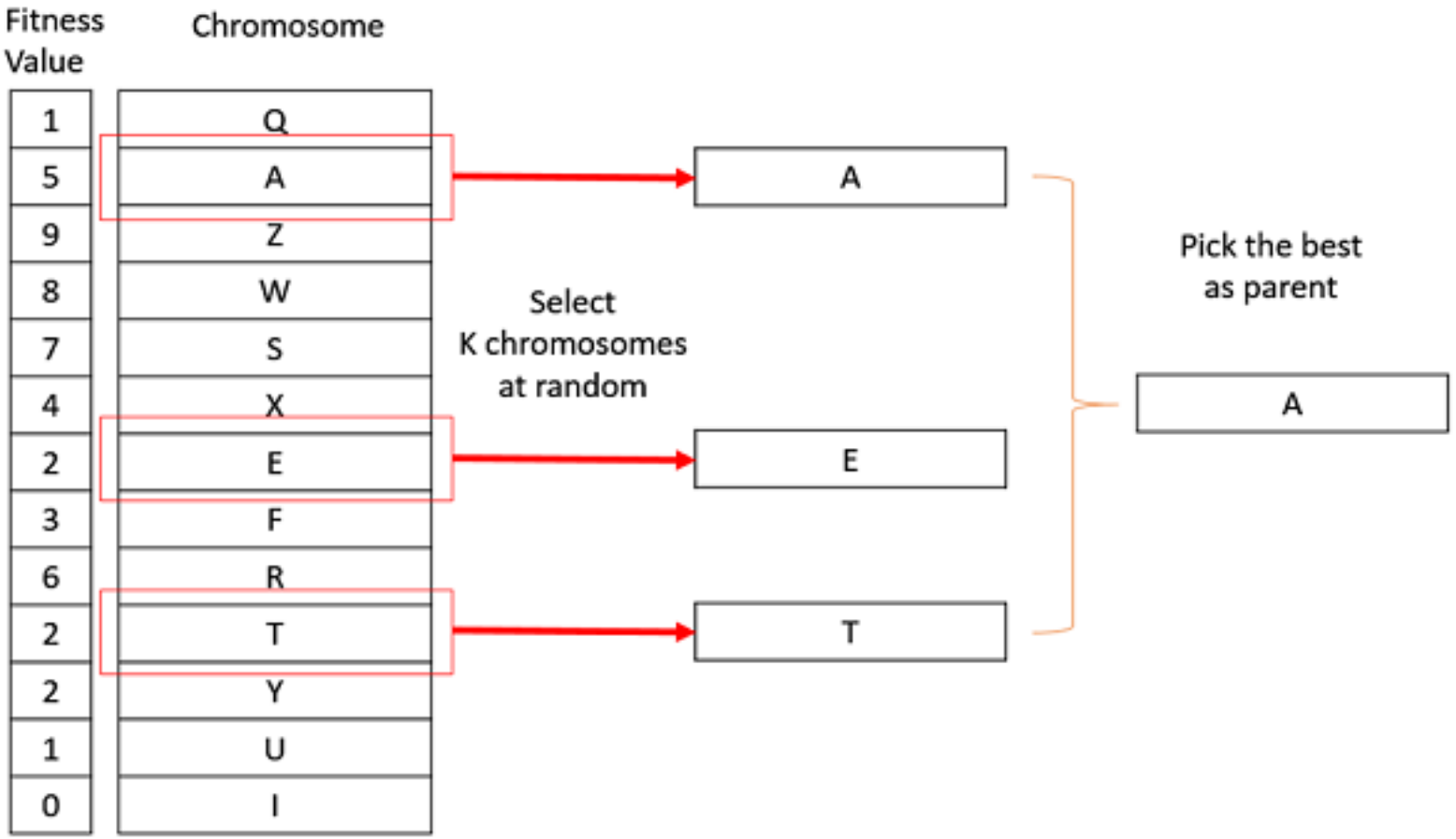
7.1.3 선택법

- 세대교체에서는 적응도가 높은 유전자를 갖는 개체를 선택하는데, 선택하는 방법은 다음과 같이 몇 가지가 있음.
  - ✓ **엘리트 보존:** 현 세대에서 적응도가 가장 높은 것은 교차, 돌연변이를 수행하지 않고 다음 세대에 그대로 물려 주는 것을 의미함.
  - ✓ **룰렛 선택:** 자식을 적응도에 비례하는 확률로 선택한다(확률은 pi=fi/F, 단 fi는 개체 i의 적응도 F=Σfi). 적응도가 낮다고 해서 버리는 것은 아니지만 실제로는 자식도 유한 개수이므로 적응도가 낮은 개체는 무시됨.
  - ✓ **토너먼트 선택:** 무작위로 선택한 개체 중(보통 2개)에서 가장 적응도가 높은 개체를 선택한다. 보통 높은 적응도를 갖는 것만 남기 때문에 수렴은 빠르지만 국소해에 빠질 가능성이 높음.
- 다른 선택법도 있으며 조합하여 사용하는 경우도 있음. 홀랜드가 최초로 제안한 방법은 **룰렛 선택**이지만 그 이후 개선을 위한 연구가 이루어지고 있음.

⊙ 룰렛 선택


wheel is rotated
5
14%
4
12%
1
31%
selection
point
3
38%
2
5%
Fittest individual
has largest share of
the roulette wheel
Weakest individual
has smallest share of
the roulette wheel

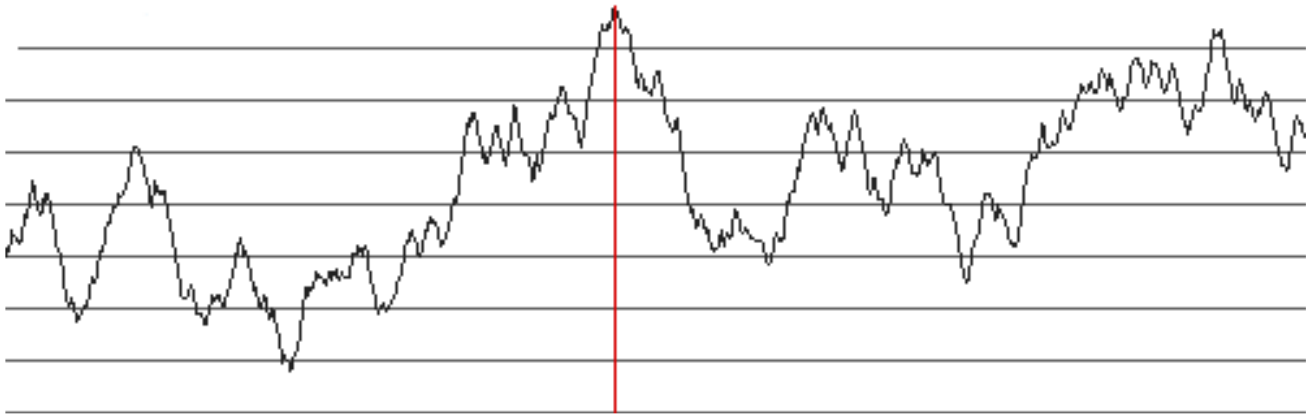
⊙ 토너먼트 선택

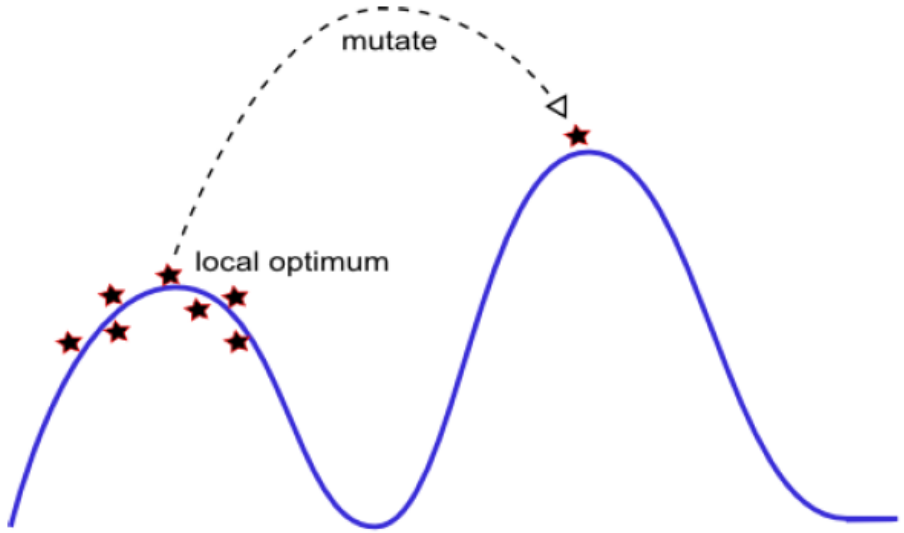

Fitness Value
Chromosome
1
Q
5
A
9
Z
8
W
7
S
4
X
2
E
3
F
6
R
2
T
2
Y
1
U
0
I
Select K chromosomes at random
A
E
T
Pick the best as parent
A

7.1.4 교차법

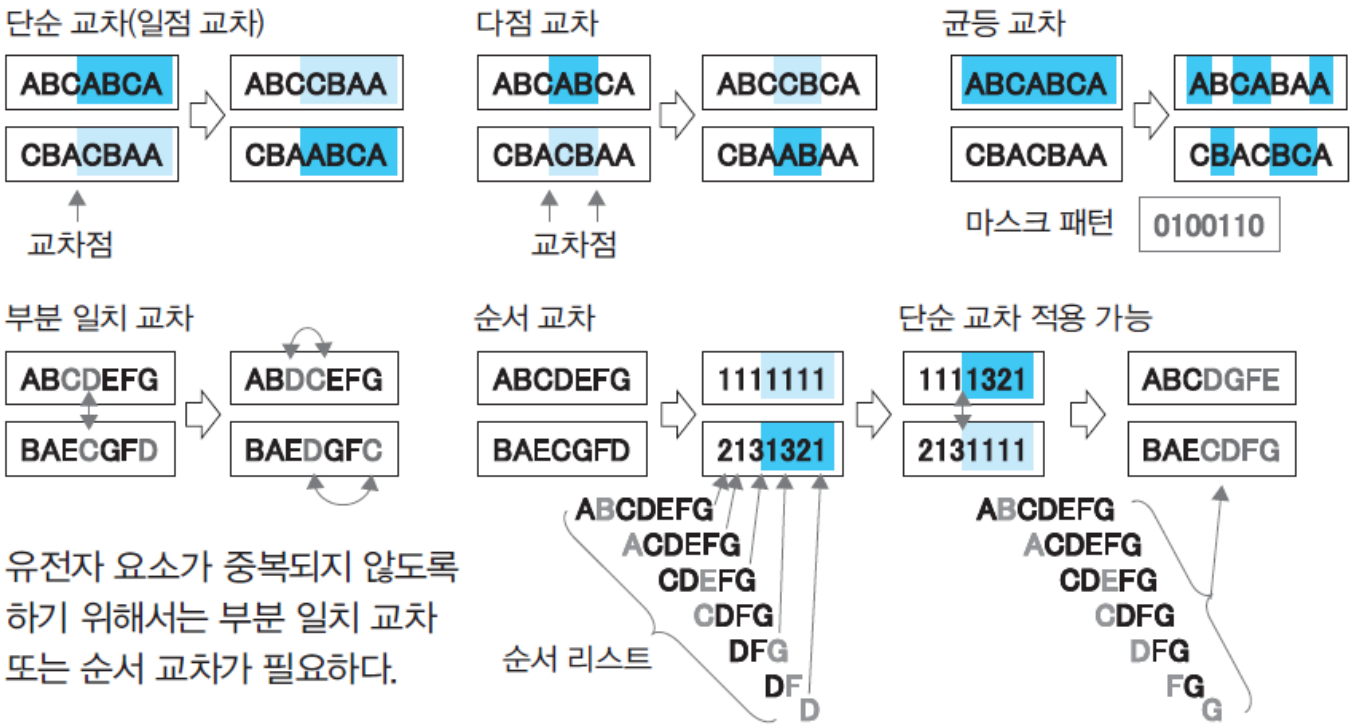
- 선택된 개체의 집단에서 2개씩 조합하여 부모 쌍을 몇 개 만들고, 부모 쌍으로 유전자 교환을 수행.
- 부모 쌍의 가장 일반적인 조합 방법은 적응도가 높은 것과 낮은 것으로 순서대로 해 나가는 것임. 이렇게 함으로써 유전자의 경향이 치우치는 것을 방지할 수 있음.
- 얼핏 보면 **적응도가 높은 것들끼리 조합**하는 편이 효율적인 것 같지만, 이럴 경우 **국소해**로 끝나 버릴 가능성이 높음.
- 오히려 낮은 적응도의 유전자 중에서 보다 좋은 해를 이끌어 낼 수 있는 요소가 숨어 있을 가능성이 있음.
- 다른 조합 방법으로는 무작위로 2개씩 쌍을 만드는 방법도 있음.
- 부모 쌍이 만들어지면 다음의 방법들로 유전자의 일부를 교환.

⊙ 국소해

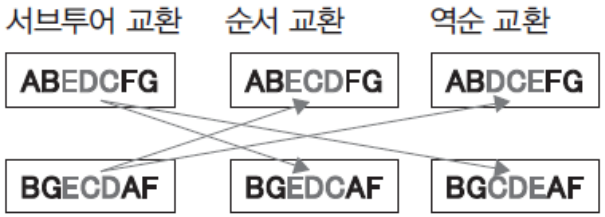




7.1.4 교차법



서브투어 교환 교차

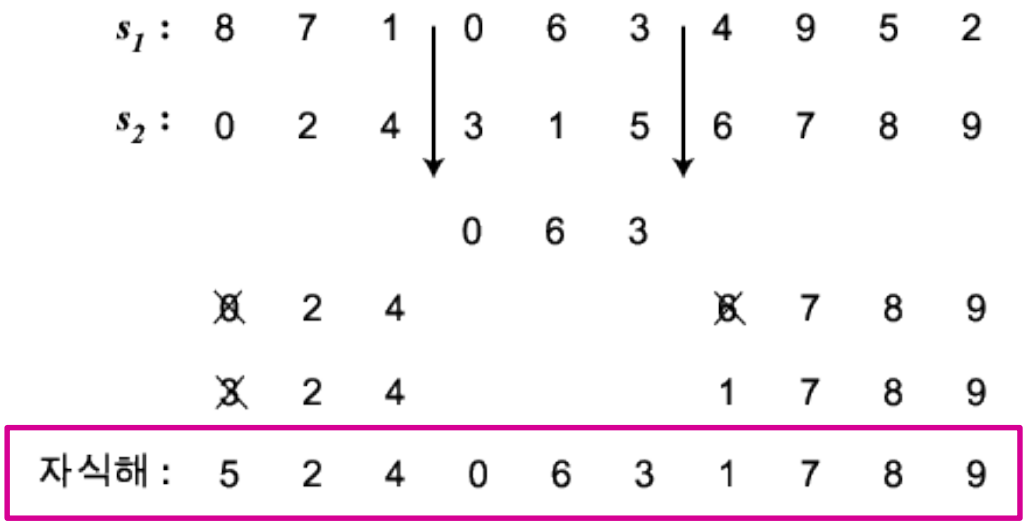


서브투어 부분을 상호 간에 그대로 교환(순서 교환)하고, 다시 역순으로 교환(역순 교환)하기 때문에 두 부모로부터 4개의 자식이 생성된다.

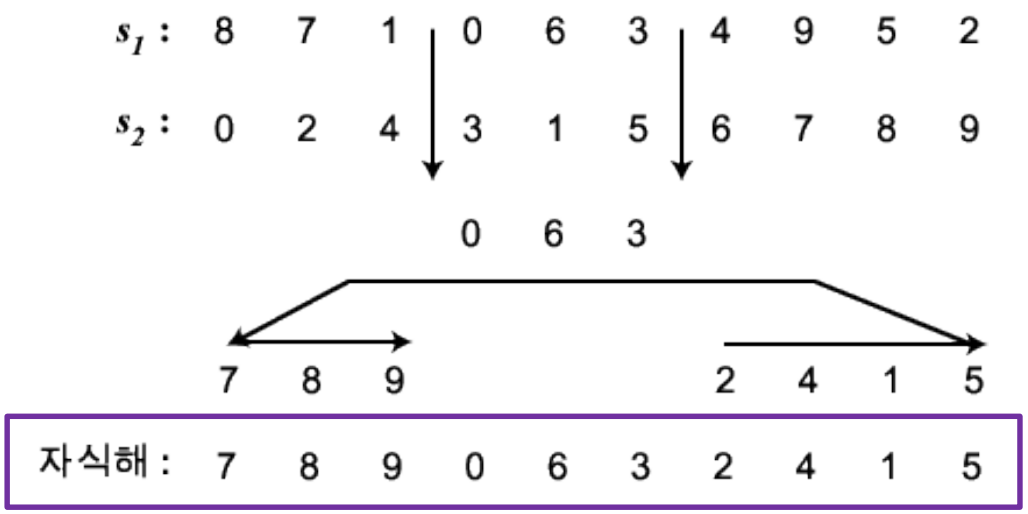
그림 4-2 교차법

7.1.4 교차법

➢ 부분 일치 교차



➢ 순서 교차



7.1.4 교차법

- **일점 교차(단순 교차):** 적당한 한 곳을 정하여 그 이후의 유전자를 교환.
- **다점 교차:** 부분적으로 여러 개의 유전자를 일제히 교환. 몇 군데라도 상관없음.
- **균등 교차:** 부모 쌍으로부터 마스크 패턴에 따라 유전자를 복사. 예를 들어, 0이면 부모 1로부터, 1이면 부모 2로부터 유전자를 복사.
- **부분 일치 교차:** 순서가 중요한 문제가 되는 경우에는 일점 교차로 교차 쌍을 정하고 각 개체 내에서 바꿔 넣음.
- **순서 교차:** 유전자의 중복을 허용하지 않는 경우에는 유전자 자체 교환이 아니고 순서대로 바꾸어 넣어 교차를 수행.
- **서브투어 교환 교차:** 복수의 개체에 공통적인 부분 또는 좋은 성질을 갖는 부분을 유지하면서 교차를 수행. **스키마**라고도 함.

7.2 유전 알고리즘의 구체적 예

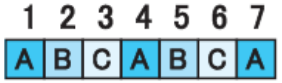
- **재산 분배 문제**
  - ✓ **수식화가 어려운 조합 최적화 문제**로서 다양한 가치를 갖는 물건을 여러 사람에게 정해진 비율로 나누어 주는 재산 분배 문제를 생각.
  - ✓ 아버지의 유언은 재산을 장남, 차남, 삼남에게 **4:2:1의 비율**로 물려준다는 것이었으며, 자산 가치가 다른 7개의 물건이 재산으로 남아 있음.

7.2 유전 알고리즘의 구체적 예

재산 분배 문제 아버지의 유언은 재산을 장남, 차남, 삼남에게 4:2:1의 비율로 물려준다는 것이었으며, 자산 가치가 다른 7개의 물건이 재산으로 남아 있다.

❶ 모델화
3명의 형제는 A, B, C. 물건의 자산 가치는 1, 2, 3, 4, 5, 6, 7억. 이것을 4:2:1로 나누는 것이 목표.

❷ 코딩
유전자는 왼쪽부터 자산 가치 순으로 소유자를 나열한 것. 적응도 평가는 A:B:C = 4:2:1과의 편차.



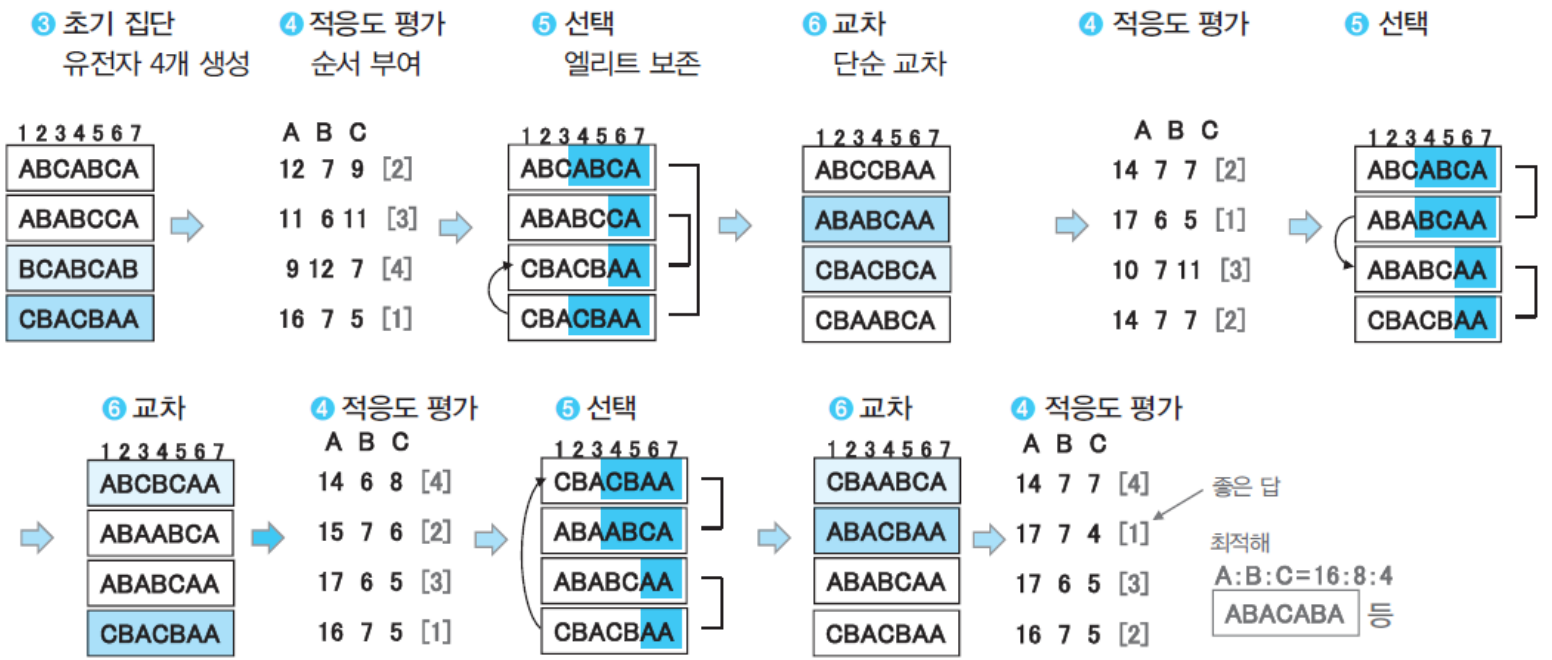


그림 4-3 재산 분배 문제

7.2 유전 알고리즘의 구체적 예

- 얼핏 보기에는 재산 총액과 각자가 취할 몫에 대한 비율로 간단하게 분배할 수 있을 것으로 여겨짐.
- 그러나 **재산 물건이 연속적으로 분할될 수 있는 형태가 아니기 때문에** 제대로 분배하려고 하면 두더지잡기 게임처럼 들쭉날쭉한 상태에 빠짐.
- 그래서 생각할 수 있는 분배 몇 가지를 시험 삼아 해 보고, **기대 비율이 가장 근접한 분배에 따르도록 함.**
- 여기서 유전 알고리즘의 적용 가치가 드러남.

7.2 유전 알고리즘의 구체적 예

- 여기서는 재산 물건 7개를 3명의 아들에게 4:2:1의 비율로 분배하는 것을 생각하였음.
- 유전자는 재산 물건별로 상속자를 나열한 7개 요소의 벡터로 하고, 적응도는 각 자식의 분배 비율과 기대 비율과의 차이로 함.
- 적당히 분배된 4가지 유전자 초기 집단으로 시작하여 선택과 교차를 반복.
- 매번 세대교체에서는 4개의 유전자를 2개씩 2조의 부모 쌍으로 하여 각 부모 쌍으로부터 2개씩, 모두 4개의 자식 유전자를 만듦.
- 이것을 여러 번 반복하면 개체로서도, 전체로서도 적응도가 점차 좋아지는 것을 알 수 있음.

7.3 재산 분배 문제 해설

- **앞의 그림에서 세대교체의 모습을 자세히 살펴보자.**

- ③의 초기 집단은 적당히 선택한 4개의 유전자로 이루어짐.
- 첫 번째 유전자는 물건 1은 A, 물건 2는 B, 물건 3은 C,..., 물건 7은 A가 상속받는 것을 나타냄.
- 두 번째 유전자를 같은 방식으로 보면 A는 물건 1, 3, 7을, B는 물건 2, 4를, C는 물건 5, 6을 상속받는 것을 나타냄.
- 세 번째 유전자는 A가 3, 6, B가 1, 4, 7, C가 2, 5를, 네 번째 유전자는 A가 3,6, 7, B가 2, 5, C가 1, 4를 상속받는 것을 나타냄.

7.3 재산 분배 문제 해설

- ④에서는 이 유전자들의 적응도를 평가하고 있음.
- 각 유전자가 나타내는 분배를 보면, 첫 번째는 A가 물건 합계 12억, B가 7억, C가 9억으로 분배되며, 두 번째는 A가 11억, B가 6억, C가 11억, 세 번째는 A가 9억, B가 12억, C가 7억, 네 번째는 16억, 7억, 5억으로 분배.
- 이 분배들이 목표인 4:2:1이라는 비율에 어느 정도 가까운지가 적응도이지만, 여기서는 엄밀한 계산은 생략하고 적응도가 높은 순으로 순서를 매김.
- 이 경우에는 4번째 유전자가 가장 적응도가 높고 이어서 첫 번째, 두 번째, 세 번째의 순서로 결정.

7.3 재산 분배 문제 해설

- ⑤에서는 선택을 수행. 엘리트 보존에 따라 가장 적응도가 낮은 유전자를 가장 높은 것으로 바꿔 놓음.
- 즉, 세 번째 유전자를 네 번째 유전자로 교체.
- 이렇게 하면 보통 적응도가 가장 높은 유전자는 2개가 됨. 그리고 이 상태로 적응도가 가장 높은 것과 가장 낮은 것을 한 쌍으로 정하고, 그 다음으로 높은 것과 낮은 것을 다시 한 쌍으로 정함.
- 이 경우에는 세 번째와 두 번째 유전자가 한 쌍(쌍 1), 네 번째와 첫 번째 유전자가 다른 한 쌍(쌍 2)이 됨.
- 여기서는 단순 교차를 수행하고 교차점을 쌍 1은 2(뒤로부터 2개 요소), 쌍 2는 4(뒤로부터 4개 요소)로 함.

7.3 재산 분배 문제 해설

- ⑥에서는 실제로 단순 교차(일점 교차)를 수행. 즉, 쌍 1은 2개 유전자의 뒷부분 2개 요소를 교환하고, 쌍 2는 4개 요소를 교환.
- ④로 돌아가 유전자 조작에 따라 생성된 4개의 자식 유전자에 대한 적응도를 평가. 그 결과, 적응도가 가장 높은 것은 두 번째 자식 유전자였으며, A, B, C의 분배가 17억, 6억, 5억이 되었음.
- 가장 낮은 것은 세 번째 자식 유전자이므로 선택(⑤)에서 세 번째를 두 번째로 교체하고, 다시 2조의 쌍을 만들어 교차점을 2와 4로 하여 교차(⑥)를 수행.
- 다시 ④로 돌아가 손자 유전자의 적응도를 평가하면 적응도가 가장 높은 것은 16억, 7억, 5억으로 처음 상태로 돌아간 것처럼 보이지만, 적응도가 가장 낮은 것도 14억, 6억, 8억이 되었음.

7.3 재산 분배 문제 해설

- 처음 상태와 비교하면 전체적으로는 비교적 적응도가 높은 것들로만 재편되었다는 것을 알 수 있음.
- 더욱이 선택(⑤)을 수행하여 가장 적응도가 낮은 첫 번째 손자 유전자를 네 번째 손자 유전자를 교체하여 쌍을 만들고, 교차(⑥)를 수행하면 가장 적응도가 낮은 첫 번째 증손자 유전자도 14억, 7억, 7억이 되므로 전체적으로도 엘리트들만의 집단이 되었음.
- 가장 적응도가 높은 두 번째 증손자 유전자는 17억, 7억, 4억이므로 목표인 4:2:1의 분배인 16억, 8억, 4억과 비교하면 A가 1억 더 받고, B가 1억 적은 결과가 되었지만 그런대로 분배가 잘 된 것으로 볼 수 있음.

7.3 재산 분배 문제 해설

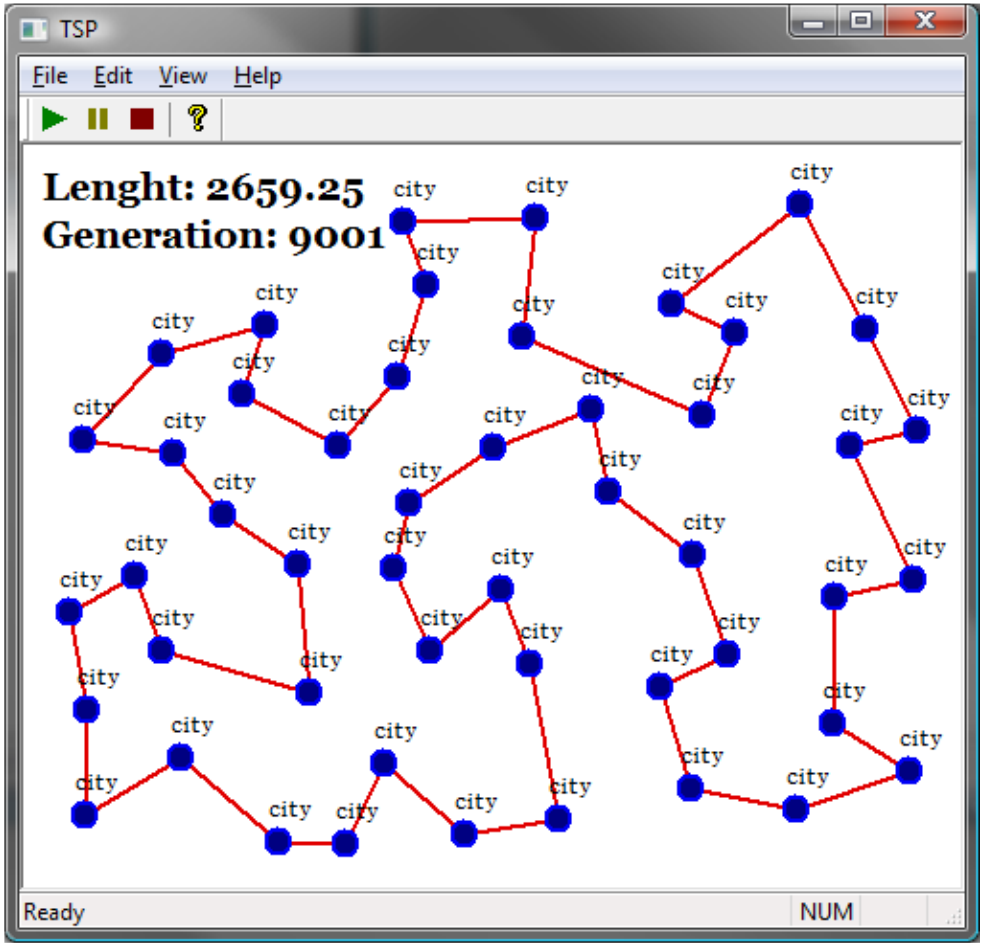
- 단, 이 이후에 몇 번 정도 더 세대교체를 수행하면 목표하는 분배가 될 수도 있겠지만 보장할 수는 없음.
- 관점을 바꾸어 보면, 엘리트들만의 유전자는 더 이상 발전할 수 없는 곳까지 와 버렸기 때문에 세대교체를 수행해도 변화가 없음.
- 그런 상태가 되었을 경우 엘리트라고 할 수 없을 정도로 적응도가 낮은 유전자를 넣어 보면, 일시적으로는 전체 적응도가 낮아지더라도 머지않아 이전보다 더 좋은 결과가 얻어질 수도 있음.
- 이 부분에 대한 이론적 근거가 부족한 것이 아쉽지만 적용해 보면 확실히 좋아지기도 함.
- 적응도 계산이 필요하지만 세대교체 자체는 동일한 순서로 반복되므로 매우 편리한 방법임.

7.4 유전 알고리즘의 응용

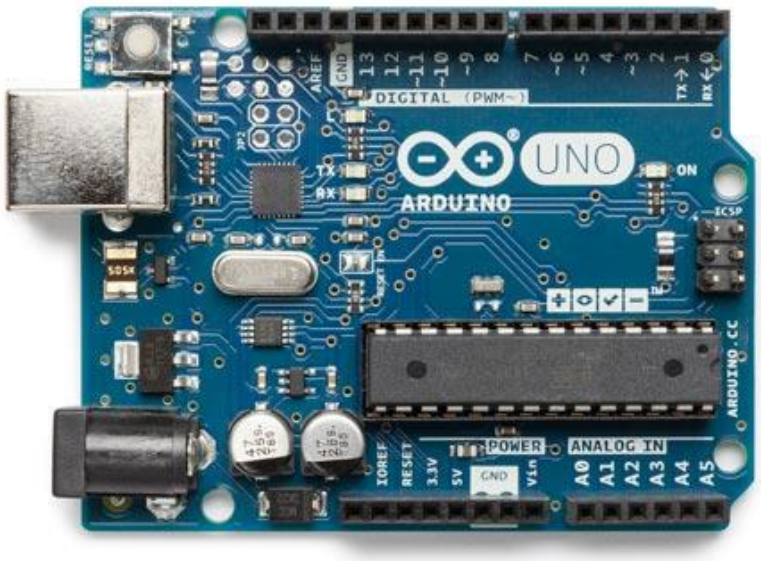
- 유전 알고리즘은 이론적 배경이 아직도 완벽하지 않지만 응용 범위는 넓음. 주요 분야는 다음과 같음.
  - ✓ **조합 최적화 문제:** 주어진 제약 조건 내에서 가장 효과적인 조합을 구함. 냅색 문제(Knapsack Problem; KP)와 순회 외판원 문제 (Traveling Salesperson Problem; TSP)가 유명함.
  - ✓ **배치 설계 문제:** 주어진 기능 블록을 작은 공간에 가장 효과적으로 배치한다. LSI 설계, 상업시설 설계 등.
  - ✓ **배치 표시 문제:** 트리 구조와 그래프 형태로 가지의 교차를 최소화 한다. 이것은 상용 소프트웨어에도 사용됨.
- 이 외에 **태스크 스케줄링**(많은 공정을 최적 순서로 진행), **제어 문제**(에어컨 온도 제어 등), **계획 문제**(버스 운행 편성, 근무 편성 등) 등의 응용 분야가 있음.

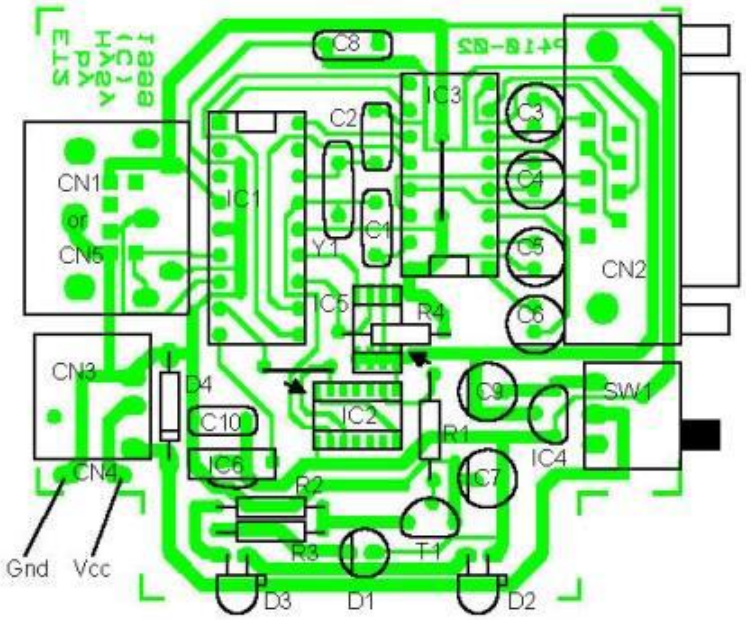
⊙ 냅색 문제 (Knapsack Problem KP)

⊙ 순회 외판원 문제 (Traveling Salesperson Problem TSP)



⊙ PCB 레이아웃 문제


DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN


C8
IC3
C2
C3
CN1
or
CN5
IC1
C1
C4
Y1
C5
CN2
IC5
R4
C6
CN3
D4
C9
SW1
C10
IC2
R1
IC4
CN4
IC6
IC7
R2
R3
T1
Gnd
Vcc
D3
D1
D2

7.4.1 조합 최적화 문제

- **조합 최적화 문제**는 2장 신경망에서도 설명하였듯이, 문제의 조합에 해당하는 수치 계산을 끝없이 반복하는 것보다 유전 알고리즘을 활용하면 더 효율적으로 좋은 해를 구하는 방법이 가능함.
- 여기에서는 2개의 유명한 문제를 살펴봄.
  - ✓ **냅색 문제:** 여러 형태, 무게의 화물들을 배낭(냅색)에 최대한 잘 집어넣는 것. '잘'이라는 의미는 제한 조건에 따라 배낭에 들어간 각 화물이 갖는 가치의 합계가 최대가 되도록 하는 것.
  - ✓ **순회 외판원 문제:** 여러 도시를 중복 없이 잘 돌아다니는 것. '잘'이라는 의미는 각 도시 간의 방문 경로에 대한 비용의 전체 합이 최소가 되도록 하는 것.

7.4.1 조합 최적화 문제

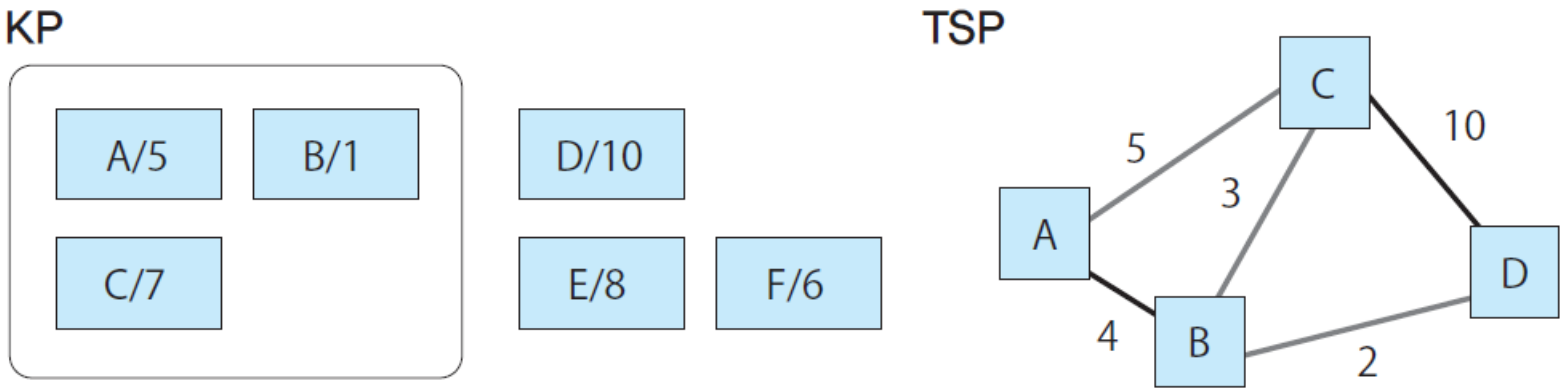


그림 4-4 냅색 문제(KP)와 순회 외판원 문제(TSP)의 개요

- 두 문제 모두 간단한 것처럼 보이지만, 가능한 방법을 모두 하나 하나 조사하려고 할 경우 **화물의 개수나 도시 수가 50개 정도가 되면 경우의 수가 1조가 넘어** 보통의 컴퓨터에서는 도저히 대응할 수 없음.
- 유전 알고리즘을 적용하면, 반드시 최적해를 구한다는 보장은 할 수 없지만, **일정 시간 내에 좋은 해(근사해)를 얻을 수는 있음**.

7.4.1 조합 최적화 문제

- 이 문제들은 보통 다음과 같이 처리.
  - ❖ **유전자 결정**
  - ✓ **KP:** 각 비트를 각 화물에 대응(1은 있음, 0은 없음)시킨 화물 개수만큼의 길이를 갖는 비트열. 예를 들어, 111000
  - ✓ **TSP:** 도시를 방문하는 순서대로 도시 이름을 나열한 기호 벡터. 단, 길이 없는 도시는이웃하게 두지 않는다. 예를 들어, ACBD, 또는 도시 이름에 대하여 그 도시를 방문하는 순서 차례를 나열한 수치 벡터. 제약 사항은 동일함(예를 들어, 1324).

7.4.1 조합 최적화 문제

- 적응도 평가
  - ✓ KP: 제약조건 내 개체는 화물 가치의 합계. 조건 외 개체는 가치를 0으로 평가
  - ✓ TSP: 도시 간 이동 거리의 총합계

- 초기 집단을 적당히 선택, 선택, 교차를 반복, 가치가 일정 수준 이상이 되면 종료
  - ✓ KP: 일반적인 선택, 교차로 충분
  - ✓ TSP: 모든 도시를 돌아다님 = 유전자의 중복이 없음 → 순서 교차 등

7.5 유전 알고리즘의 사례 및 활용 분야

- 유전 알고리즘 기초
  - www.youtube.com/watch?v=94p5NUogClM
  - www.youtube.com/watch?v=Yr_nRnqeDp0&1s
- 유전 알고리즘을 이용한 그림 그리기
  - www.youtube.com/watch?v=dO05XcXLxGs&t=33s
  - www.youtube.com/watch?v=Yz9MuI-tkiw&t=2s
  - www.youtube.com/watch?v=lpRk4FTLJXc
- NEAT(Neuro Evolution of Augmenting Topologies) 알고리즘
  - www.youtube.com/watch?v=C_VUtPNFqYw
  - www.youtube.com/watch?v=qv6UVOQ0F44
  - www.youtube.com/watch?v= 8V2sX9BhAW8

⊙ Python 또는 C/C++을 이용한 재산 분배 문제 실습

- **목표**

  ❖ 앞에서 나온 재산 분배 문제를 Python 또는 C/C++을 이용하여 작성해 보고 그 성능을 평가

  ✓ 재산 물건의 수(100개)

  → properties100.txt

  ✓ 분배할 숫자 : 5, 10, 15의 세 경우에 대하여 2,000 세대까지 실험

  → divide5.txt, divide10.txt, divide15.txt(비율(0.0 ~ 1.0)로 제공)

  ✓ 성능 평가 방법 : $\sum_{k=0}^{n-1} |target_k - eachsum_k|$ 을 최소화(n : 분배할 숫자, target : 각 자식의 목표값, eachsum : 각 자식에게 배분된 재산의 합)

⊙ Python 또는 C/C++을 이용한 재산 분배 문제 실습

- **구현 방법**
  - ❖ 유전자 집합은 리스트의 리스트(2차원 배열)로 생성
    - ✓ 재산 물건의 수(100개) → [3, 2, 2, 1, 0, 3, 4, 3, ……., 0, 1, 4]
    - ✓ 전체 유전자 세트 → [[3,,,1], [0,…2], [4,…2],…..,[1,…3]]
  - ❖ 룰렛휠 선택 기법 또는 유사 선택 기법 사용
  - ❖ 단순 교차 기법 사용
  - ❖ 엘리트 보존 기법 사용
  - ❖ 세대 별로 최고 적응도 값 출력
  - ❖ 종료 후 최고 적응도, 그 때의 배분값, 배분 결과 출력