로컬 탐색

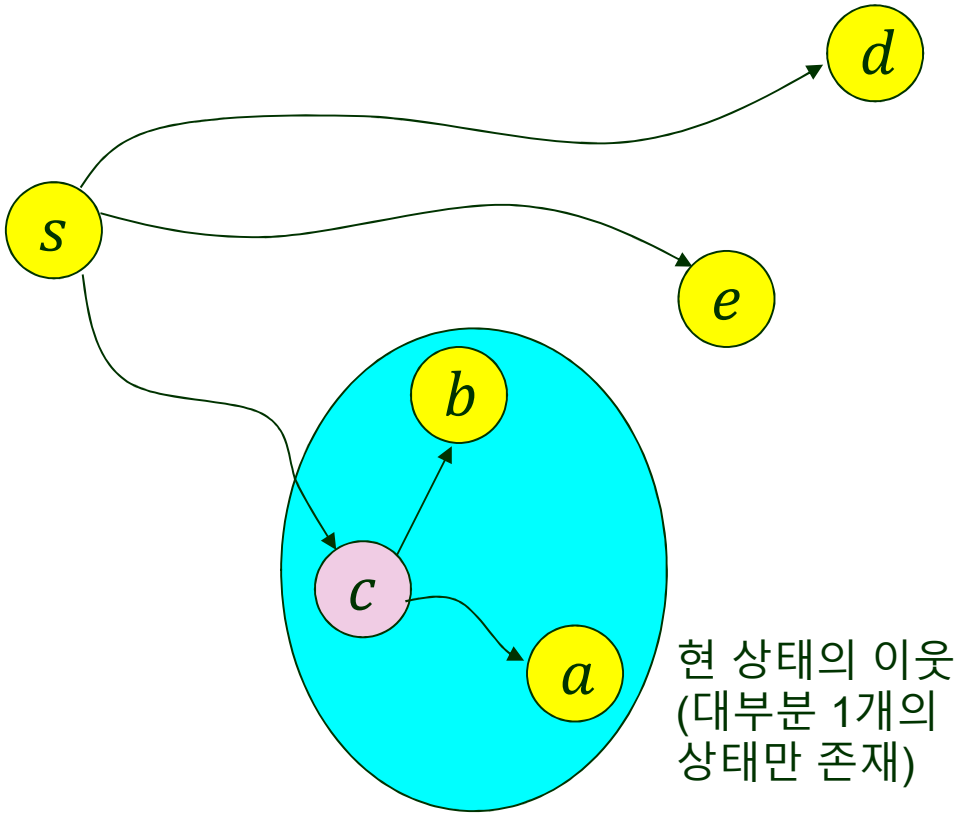
초기 상태부터 모든 경로를 체계적으로 탐험하는 것이 아니고, 현재 상태를 평가해 보고 적당한 액션을 통해 상태 변경

개요

I. 언덕 오르기
(Hill Climbing)

II. 가상 담금질
(Simulated annealing)

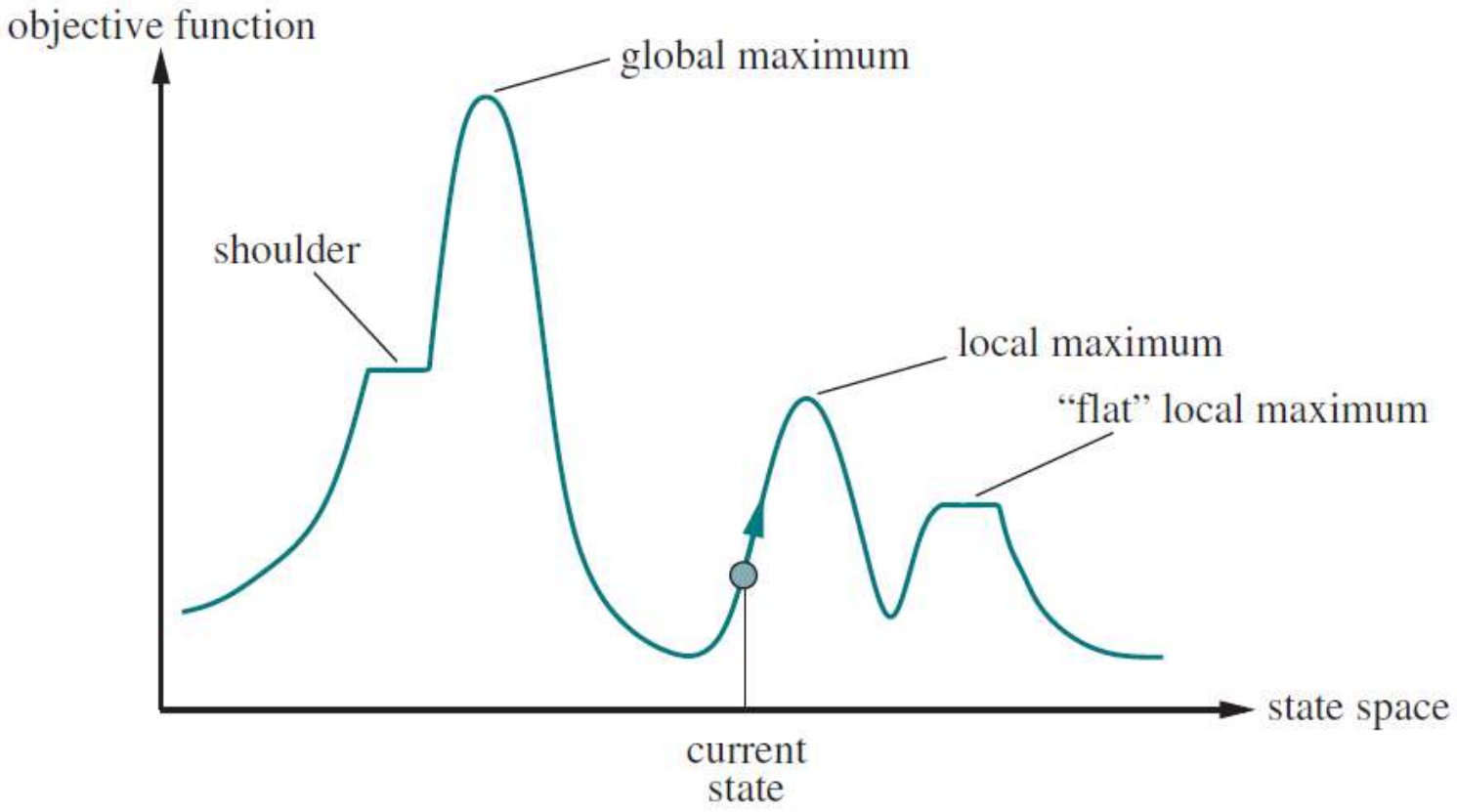
III. 유전자 알고리즘
(Genetic algorithms)



로컬 탐색의 장점

- 필요한 메모리 양이 적다.
- 체계적인 탐색에 적합하지 않은 상태 공간에서 좋은 해결책 제시 가능
- 순수 최적화 문제에 유용함 (예. 경사 하강법, 등)

상태 공간 환경


objective function
global maximum
shoulder
local maximum
"flat" local maximum
state space
current
state

I. 언덕 오르기 알고리즘

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum
  current ← *problem*.INITIAL
  while *true* **do**
    neighbor ← a highest-valued successor state of *current*
    if VALUE(*neighbor*) ≤ VALUE(*current*) **then return** *current*
    current ← *neighbor*

동률인 후임이 있을 때는
난수로 아무거나 선택

$h = 17$

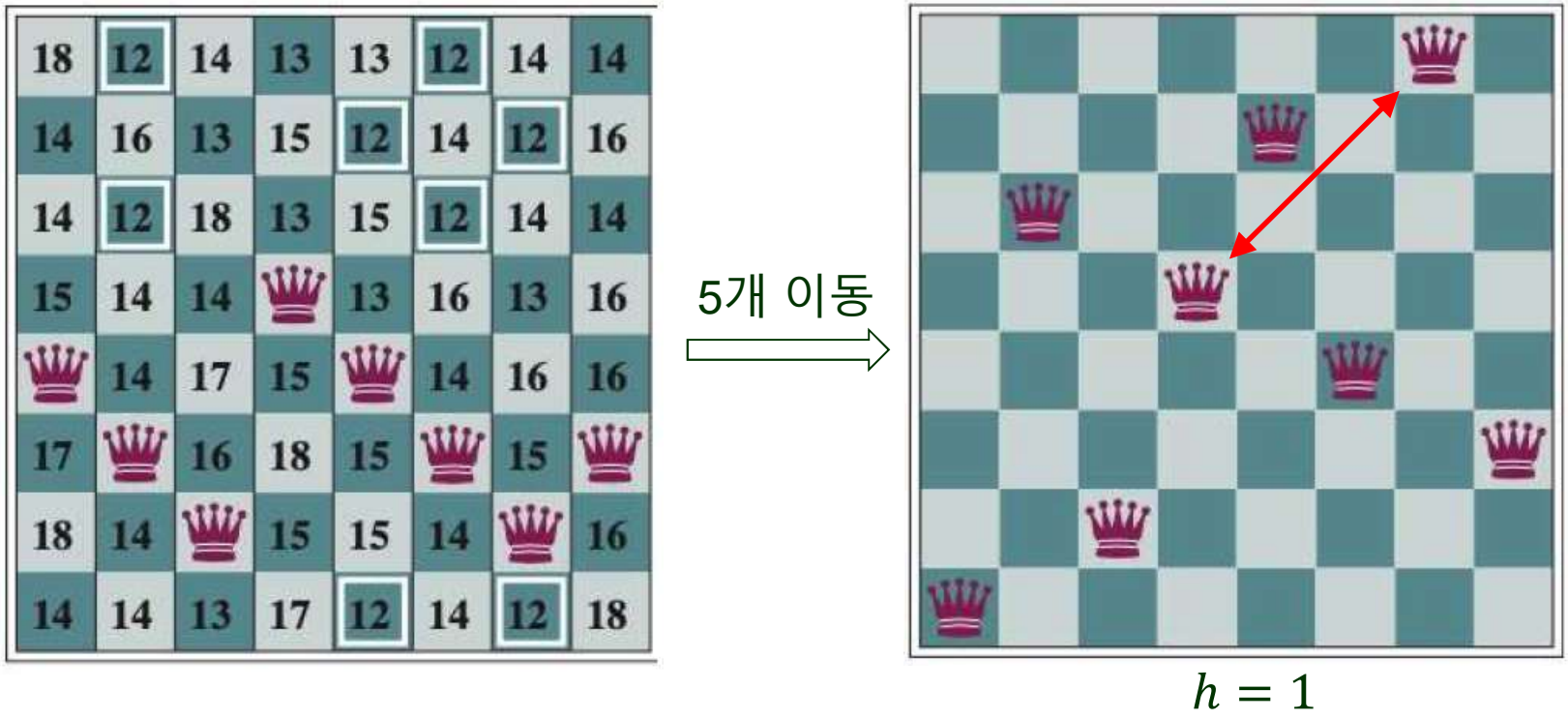
8-퀸 문제

- $h(s)$ = 상태 s에서 상호 공격하는 퀸 쌍의 개수
- 후임은 같은 열 내에서 퀸을 다른 곳으로 움직여서 만든 상태
- 최선의 후임 자리는$h(s) = 12$

  그림에서는 8개의 후임 자리가 있음

  언덕 오르기 알고리즘에서는 이 중 임의로 하나 선택

효율성 ?



- 언덕 오르기 알고리즘은 신속하게 해를 향해 진행한다는 장점

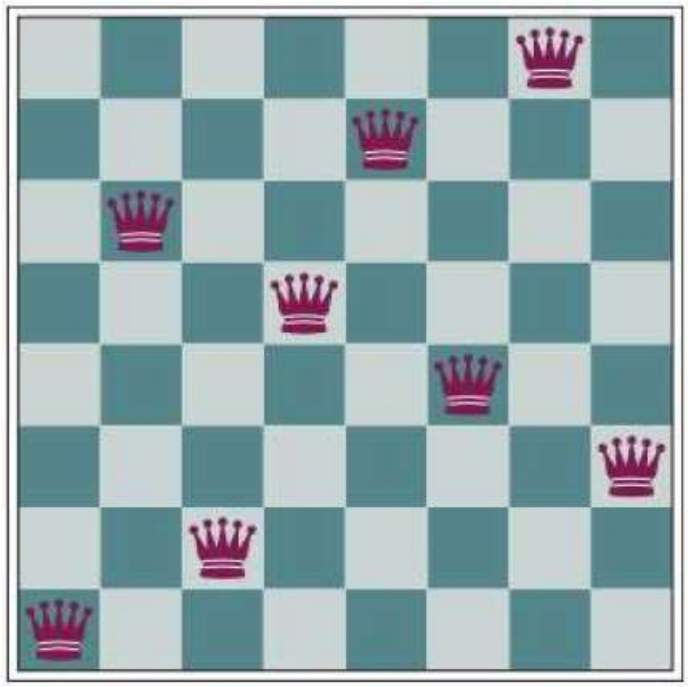
언덕 오르기 알고리즘의 단점 (1)

더 높은 값을 보유한 이웃 노드가 존재하지 않는 정점에 도달하면 종료

♠ 로컬 최대 상태 ⟶

글로벌 최대 상태는 아님

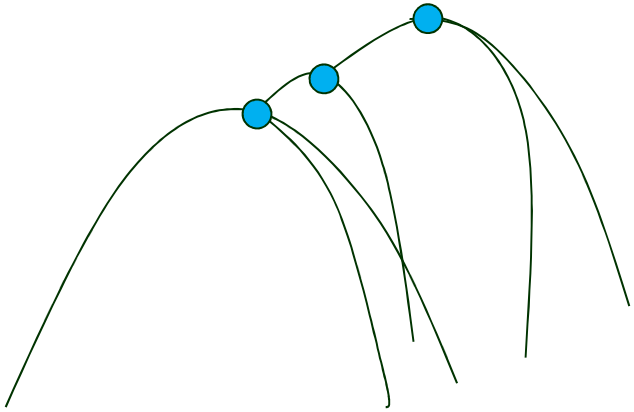
국소 최대값 근처에서 언덕 오르기 알고리즘을 적용하면 그 최대값으로 끌려가서 갇히게 될 수 있다.



퀸을 움직일 수록 충돌이 더 나는 상태임

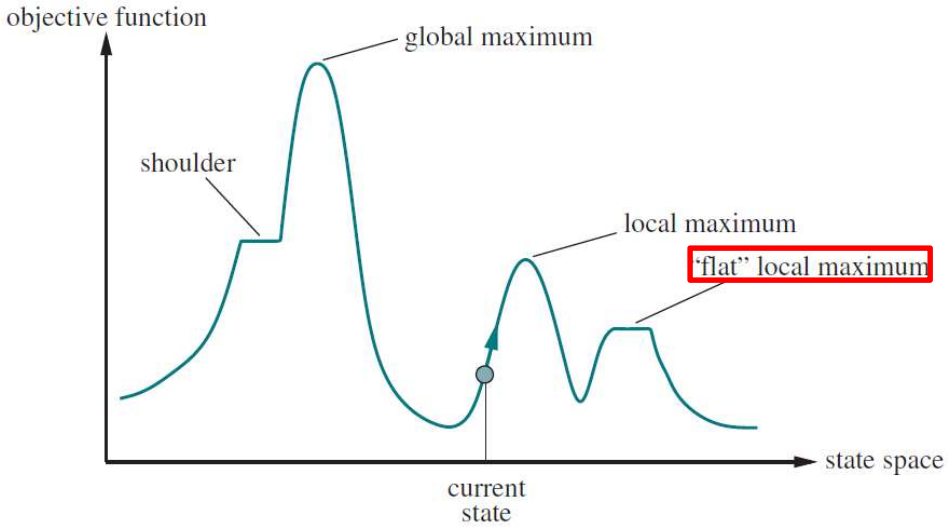
언덕 오르기 알고리즘의 단점 (2)

♠ 능선: 로컬 최대가 연속으로 존재할 경우 탐색이 어려움



각 로컬 최대 지점에서 바라볼 때 가용한 액션은 언덕 아래로 내려가는 것 뿐

♠ 고원 지대: 올라가는 액션이 존재하지 않음

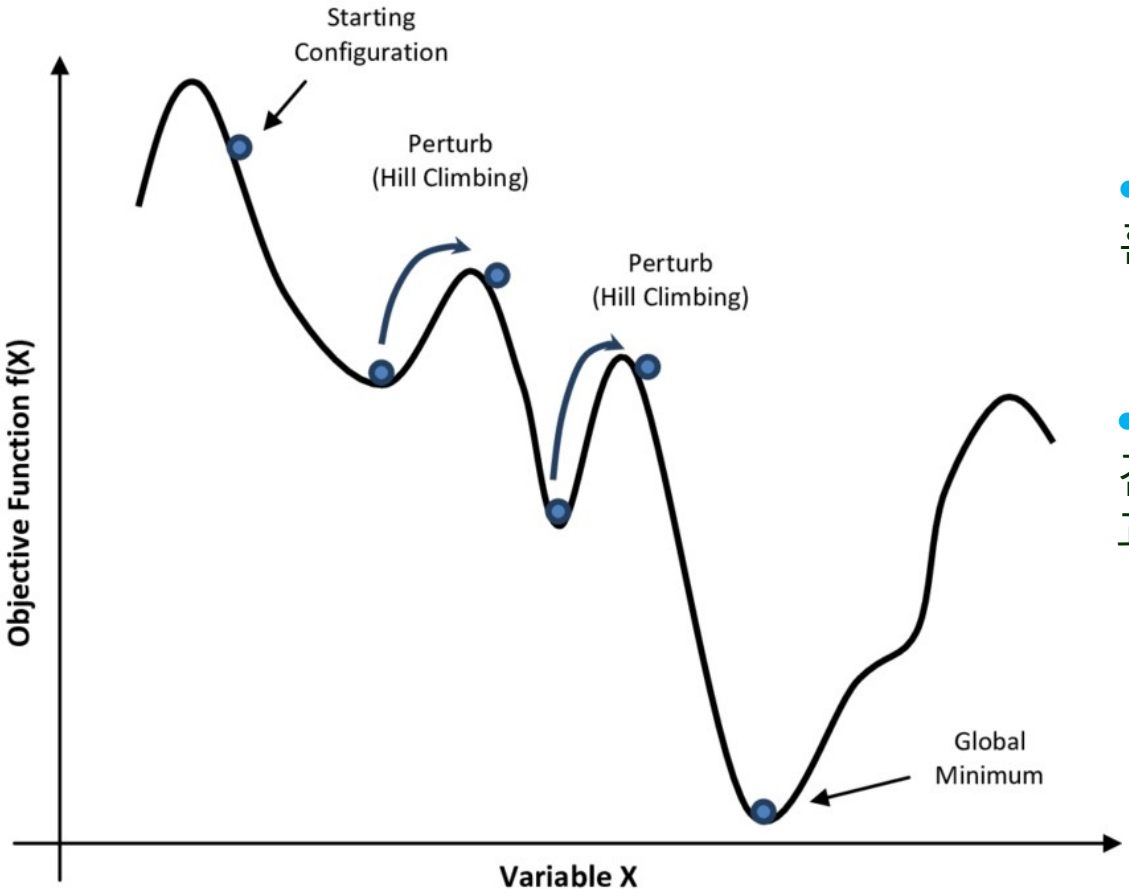


언덕 오르기 알고리즘의 변종

- 통계적 언덕 오르기 알고리즘
  - Random selection among the uphill moves.
  - Probability of selection varying with steepness..
- 최초 선택 언덕 오르기 알고리즘
  - 현재 상태보다 더 나은 후계자가 발견될 때까지 무작위로 후계자를 생성
  - 많은 후계자가 존재하거나 목적 함수를 평가하는 데 비용이 많이 들 때 유용
- 무작위 재시작 언덕 오르기 알고리즘
  - 무작위로 초기 상태로 돌아간 후 검색 다시 시작

II. 모의 정련 알고리즘

정련(담금질): 금속을 고온으로 가열한 후 천천히 냉각시켜서 낮은 에너지 결정 상태에 도달하도록 함으로써 강화시키는 과정



- 시작 단계에서는 강하게 흔들어 준다(고온 가열에 해당)
- 점차 흔들기 강도를 줄여 간다(온도를 천천히 낮춰주는 과정에 해당)

모의 정련 알고리즘

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  current ← *problem*.INITIAL
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow$ VALUE(*current*) – VALUE(*next*)
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{-\Delta E/T}$

최소값 찾기

온도 → T

$\lim_{t\to\infty} T = 0$

해를 발견함

나쁨 정도 → ΔE

다음 상태가 더 좋으므로 그쪽으로 이동

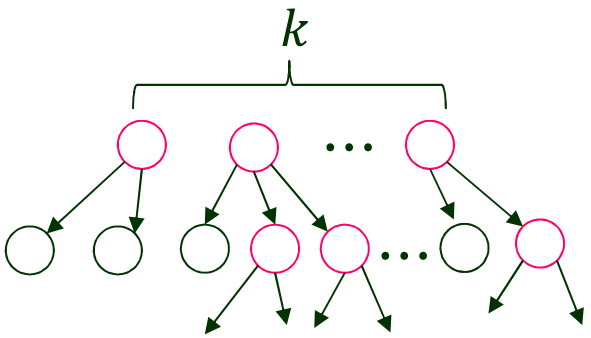
- 무작위로 뽑은 다음 상태가 개선 효과가 있으면 ($\Delta E > 0$) 그쪽으로 이동
- 개선 효과 없으면 일정 확률(지수적으로 감소하는)을 적용하여 이동
  - 움직임에 대한 나쁨 정도 ΔE 감소
  - 온도 T를 내려서 식히는 과정과 유사

나쁜 이동은 큰 T값을 갖는 시작 시점에 더 많이 발생한 후 T값이 감소함에 따라 점차 감소

- 나쁜 움직임을 허용하여 로컬 최소지점을 탈출

로컬 빔 탐색 (국소 다발 검색)

하나가 아니라 k 개의 상태를 추적



1.무작위로 생성한 k 개의 상태로 시작

2. 생성된 k 개의 상태에 대한 후임 상태 생성

3. 후임이 목표 상태이면 중지

4. 아니면 k 개의 최선의 후임을 유지한 채로 2.번 단계로 이동

♣ k 개의 상태들 간에 다양성이 결여될 가능성 존재
(비슷한 k개의 상태는 1개 상태와 큰 차이가 없어 보임)

해결 방법: 확률론적 빔 검색 (후임 중 최상의 k개를 선택할 때 선택 확률을 해당 후임의 값에 비례하는 값으로 설정 (적합할 수록 더 많이 선택됨)

진화 알고리즘

- *유전 알고리즘* 이라고도 부름
- 생물학의 자연 선택설에서 영감을 받음

1. 무작위로 생성한 k개의 상태들(개체들)의 군집으로 시작
2. *가장 적합한* 개체들을 선택하여 다음 세대에 대한 부모로 지정
3. 각 ρ 개의 부모를 결합하여 자식 개체를 형성 (보통 $\rho = 2$로 지정)

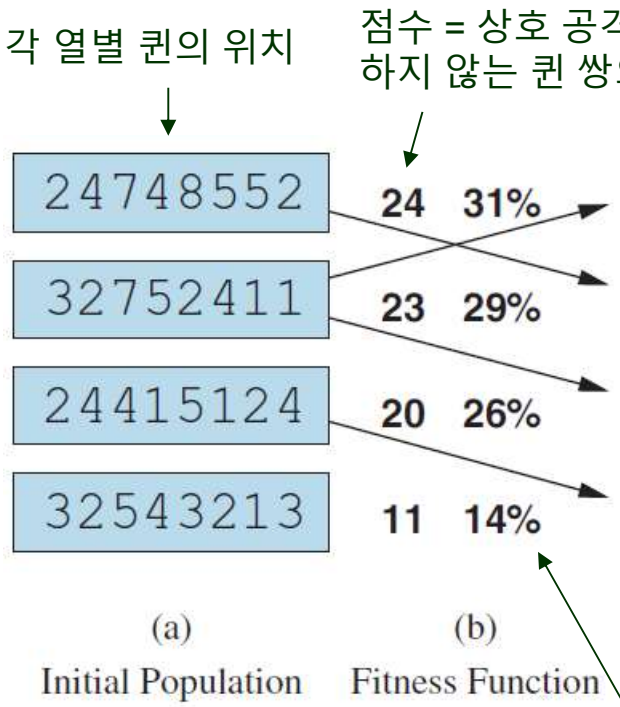
   교차: 각각의 부모 스트링을 나누어 두개의 자식 노드 생성을 위해 재결합 시킴

   돌연변이: 자손의 비트들을 무작위로 변경

   컬링: 한계치 이하인 개체들은 모두 군집으로부터 버려짐

4. 2번 단계로 돌아가서 충분히 적합한 상태가 발견될 때 까지 반복
   (최선의 노드가 해로 선택될 때 까지)

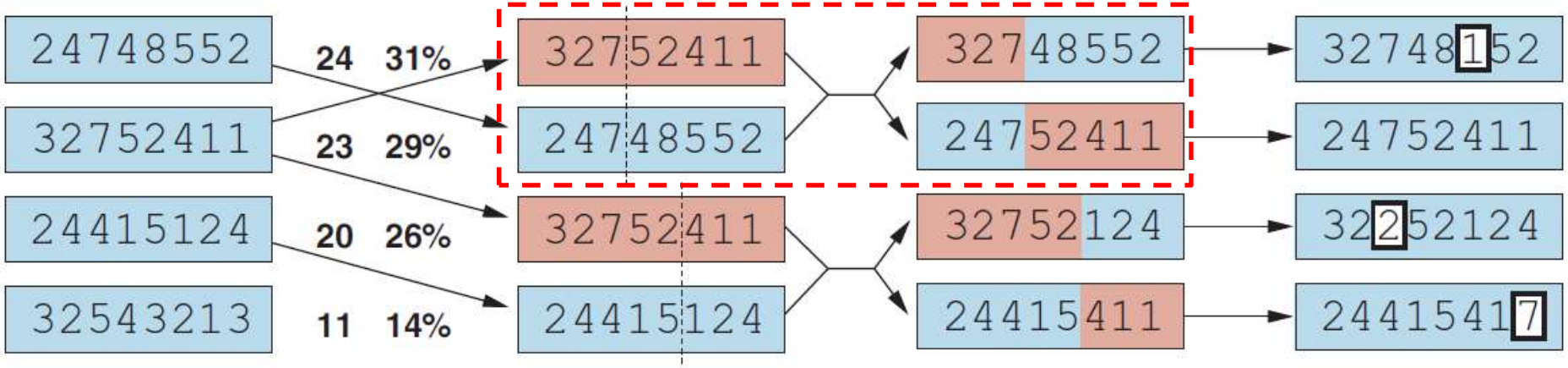
8 퀸 문제에 유전 알고리즘 적용



$$14\% = \frac{11}{24 + 23 + 20 + 11}$$

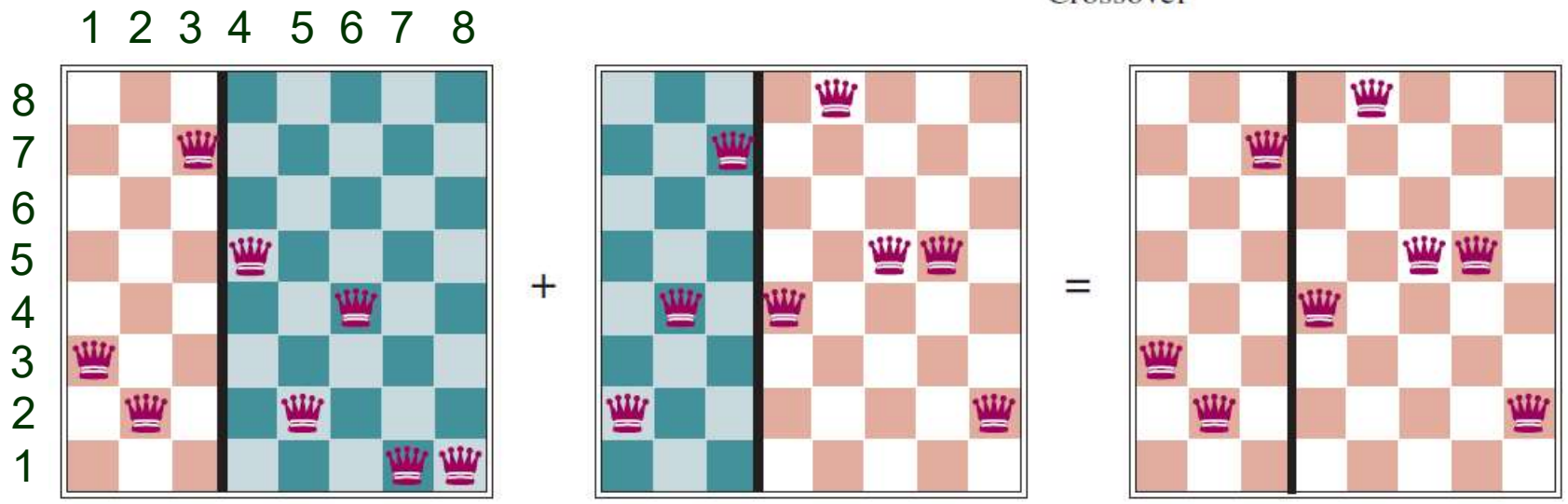
8번 열의 퀸을 무작위로
7번 위치로 이동

교차



(d)

Crossover



유전 알고리즘 적용 분야

- 복잡한 구조를 갖는 문제

  회로 설계, 공장 일정 관리

- 심층 신경망의 구조를 발전시키는 문제
- 하드웨어 오류 찾는 문제
- 분자 구조 최적화
- 이미지 프로세싱
- 학습하는 로봇, 등