문제의 종류


사람이 살면서 눈앞의 이익만 좇으면 대개 결과가 안 좋지.
알고리즘도 그런가요?
대체로 그런데, 일부 알고리즘은 눈앞의 이익만 좇아가면 최적해를 구하게 되는 경우가 있긴 하지. 이미 배운 것 중에 그런 예가 있어.
최소 신장 트리를 구하는 알고리즘은 둘 다 그런 것 같고, 최단 경로를 구하는 다익스트라 알고리즘도?
빙고

문제의 종류

풀 수 없는 문제들

(Unsolvable)

(Undecidable)

정지 문제

힐버트의 10번째 문제

...

여기에 속할 것이라고
강력히 추정!

Presburger 산술

...

NP-완비 문제들

현실적인 시간내에
풀 수 없는 문제들

풀 수 있는 문제들

(Solvable)

(Decidable)

최소 신장 트리 문제

최단 거리 문제

...

현실적인 시간내에
풀 수 있는 문제들

문제의 종류

- Unsolvable problems
- Solvable problems
  - Polynomial time problems(P)
  - Nondeterministic-polynomial time problems(NP)
  - P ⊆ NP
- P
  - $O(n^p)$

Asymptotic Notations for Complexity

$O(\,g(n)\,)$

– 기껏해야 $g(n)$의 비율로 증가하는 함수

$\Omega(\,g(n)\,)$

– 적어도 $g(n)$의 비율로 증가하는 함수
– $O(\,g(n)\,)$과 대칭적

$\Theta(\,g(n)\,)$

– $g(n)$의 비율로 증가하는 함수
– $\Theta(\,g(n)\,) = O(\,g(n)\,) \cap \Omega(\,g(n)\,)$

$o(\,g(n)\,)$

– $g(n)$보다 느린 비율로 증가하는 함수

$\omega(\,g(n)\,)$

– $g(n)$보다 빠른 비율로 증가하는 함수
– $o(\,g(n)\,)$과 대칭적

- $O(\ g(n)\ )$
  - Tight *or* loose upper bound
- $\Omega(\ g(n)\ )$
  - Tight *or* loose lower bound
- $\Theta(\ g(n)\ )$
  - Tight bound
- $o(\ g(\mathrm{n})\ )$
  - Loose upper bound
- $\omega(\ g(n)\ )$
  - Loose lower bound

Greedy Algorithms

- 눈앞의 이익만 취하고 보는 알고리즘
- 현재 시점에 가장 이득이 되어 보이는 해를 선택하는 행위를 반복한다
- 대부분 최적해와는 거리가 멀다
- 드물게 최적해가 보장되는 경우도 있다

do {

우선 가장 **좋아 보이는** 선택을 한다

} **until** (해 구성 완료)

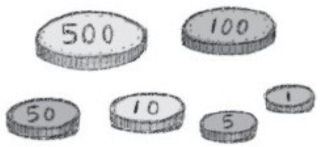
Greedy Algorithm의 전형적 구조

```
Greedy(C)
// C: 원소들의 총집합
{
        S ← Ø;
        while (C≠Ø and S는 아직 온전한 해가 아님) {
                x ← C에서 가장 좋아 보이는 원소;
                집합 C에서 x 제거; // C ← C-{x}
                if (S에 x를 더해도 됨) then S ← S ∪{x};
        }
        if (S가 온전한 해임) then return S;
                        else return "no solution!";
}
```

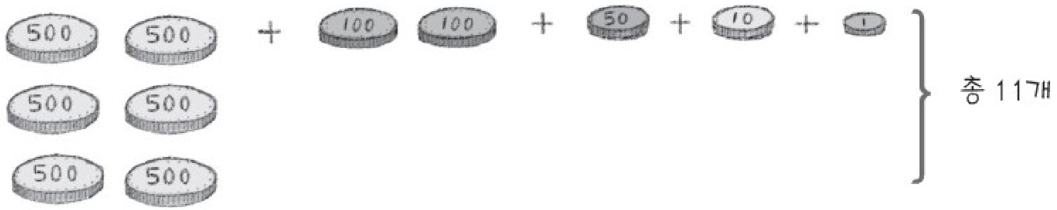
Greedy Algorithm으로 최적해가 보장되지 않는 예

동전 바꾸기

동전의 액면



3,256원 만들기

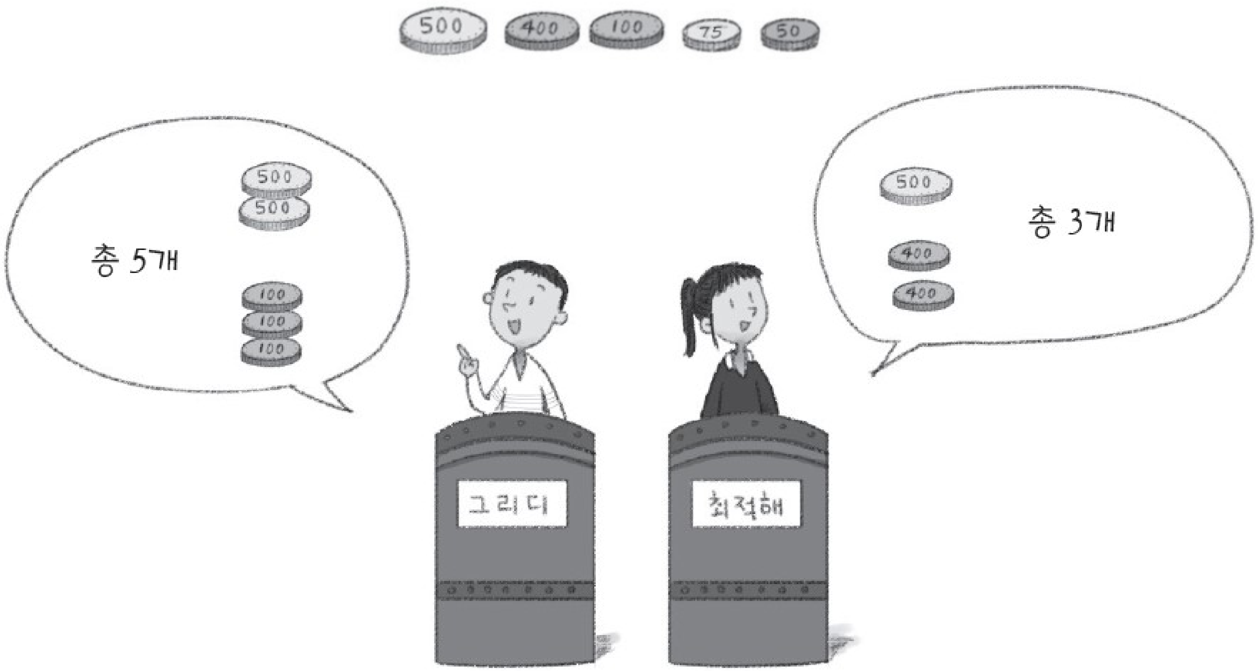


이렇게 동전의 액면이 모두 바로 아래 액면의 배수가 되면 그리디 알고리즘으로 최적해가 보장된다

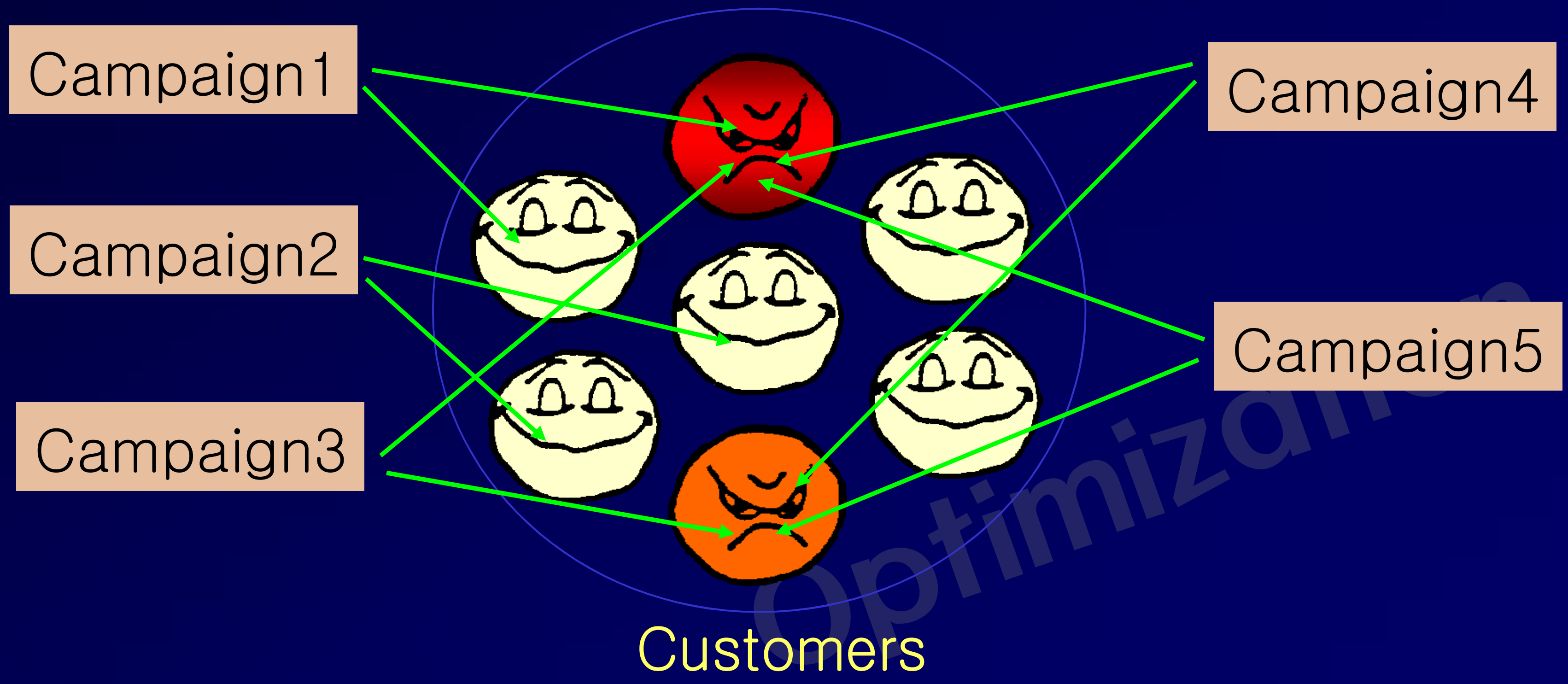
액면이 바로 아래 액면의 배수가 되지 않으면 그리디 알고리즘으로 최적해가 보장되지 않는다.

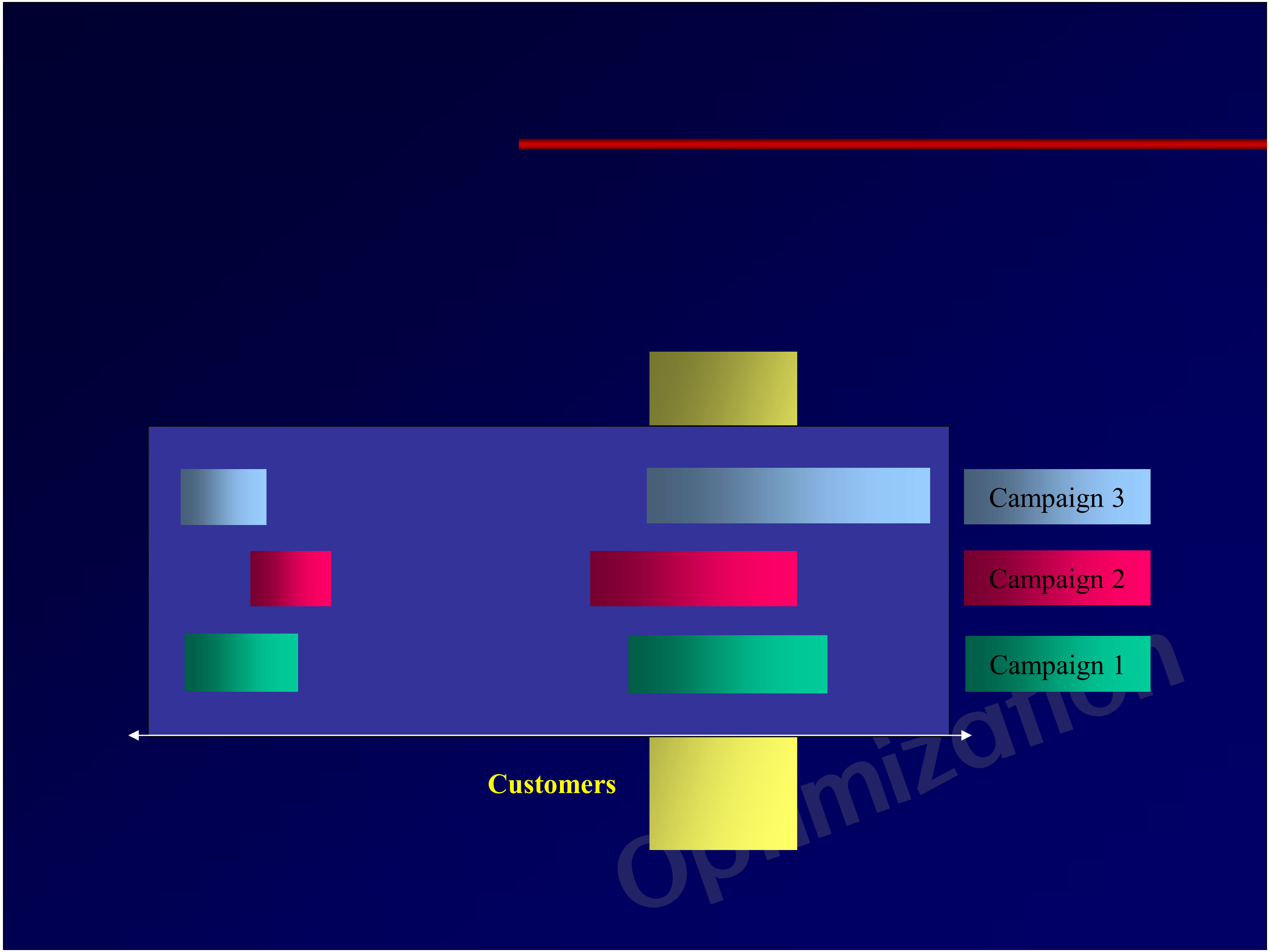
예: 다음 페이지

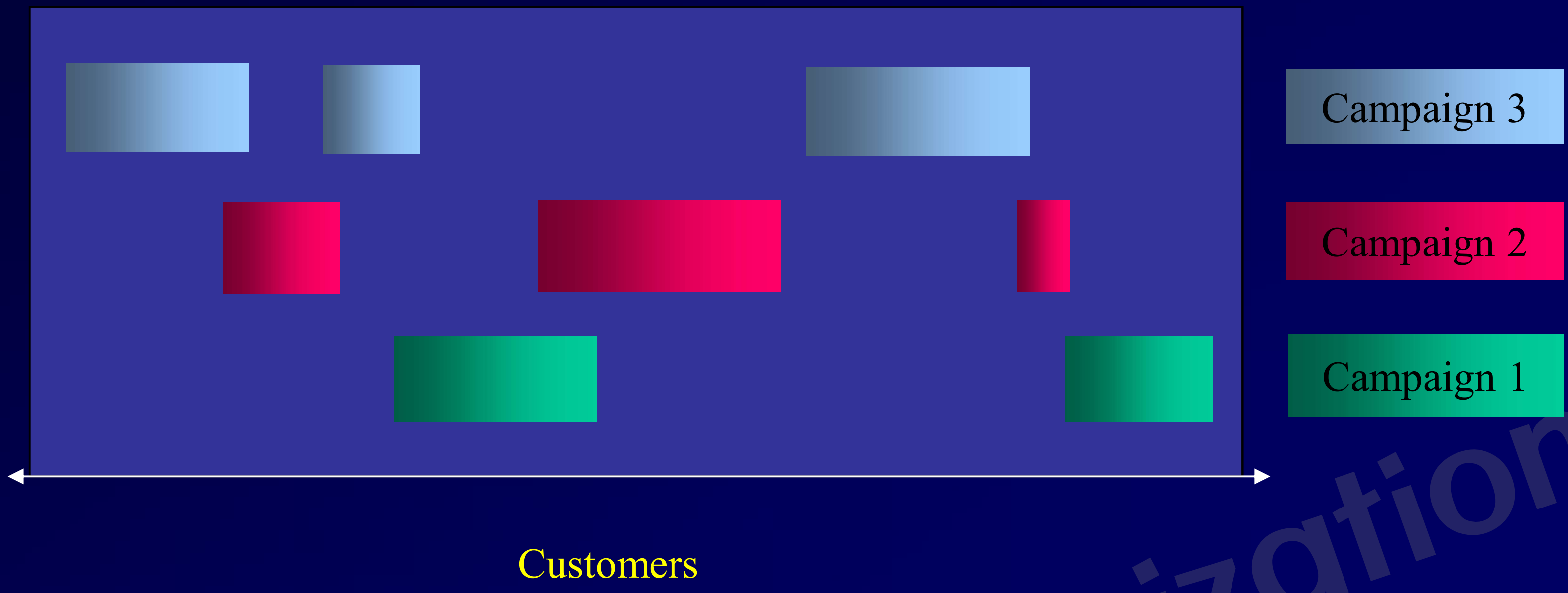
액면이 바로 아래 액면의 배수가 되지 않으면 greedy algorithm으로 최적해가 보장되지 않는다



Multi-Campaign




Campaign 3
Campaign 2
Campaign 1
Customers
Optimization



- What if 50 campaigns/week ?

Optimization

Greedy Algorithm이 최적해를 보장하는 예

- 최소 신장 트리 찾기를 위한 Prim 알고리즘과 Kruskal 알고리즘
- 최단경로를 위한 Dijkstra 알고리즘

```
Prim (G, r)
{
    S ← Φ ;
    정점 r을 방문되었다고 표시하고, 집합 S에 포함시킨다;
    while (S≠V) {
        S에서 V-S를 연결하는 간선들 중 최소길이의 간선 (x, y) 를 찾는다; ▷ (x∈S, y∈V-S)
        정점 y를 방문되었다고 표시하고, 집합 S에 포함시킨다;
    }
}
```

greedy한 부분

Greedy Algorithm이 최적해를 보장하는 예 2

회의실 배정 문제

- 회의실 1개
- 여러 부서에서 회의실 사용 요청
  - 회의 시작 시간과 종료 시간을 명시해서 신청

- Greedy한 아이디어들
  - 소요 시간이 가장 짧은 회의순 배정
  - 시작 시간이 가장 이른 회의순 배정
  - 종료 시간이 가장 이른 회의순 배정

이것만이 최적해를 보장한다

Matroid

그리디 알고리즘으로 최적해가
보장되는 수학적 구조

- Matroid 구조를 가지면 greedy algorithm으로 최적해가 보장된다

[definition] Matroid

Finite set S의 subset들의 집합인 I(즉, $I \subseteq 2^S$)가 다음 성질을 만족하면 matroid라 한다.

1. $A \in I$이고 $B \subseteq A$이면 $B \in I$이다 (heredity)
2. $A, B \in I$이고 $|A| \subseteq |B|$이면 $A \cup \{x\} \in I$인 $x \in B - A$가 존재한다
   (augmentation or exchange)

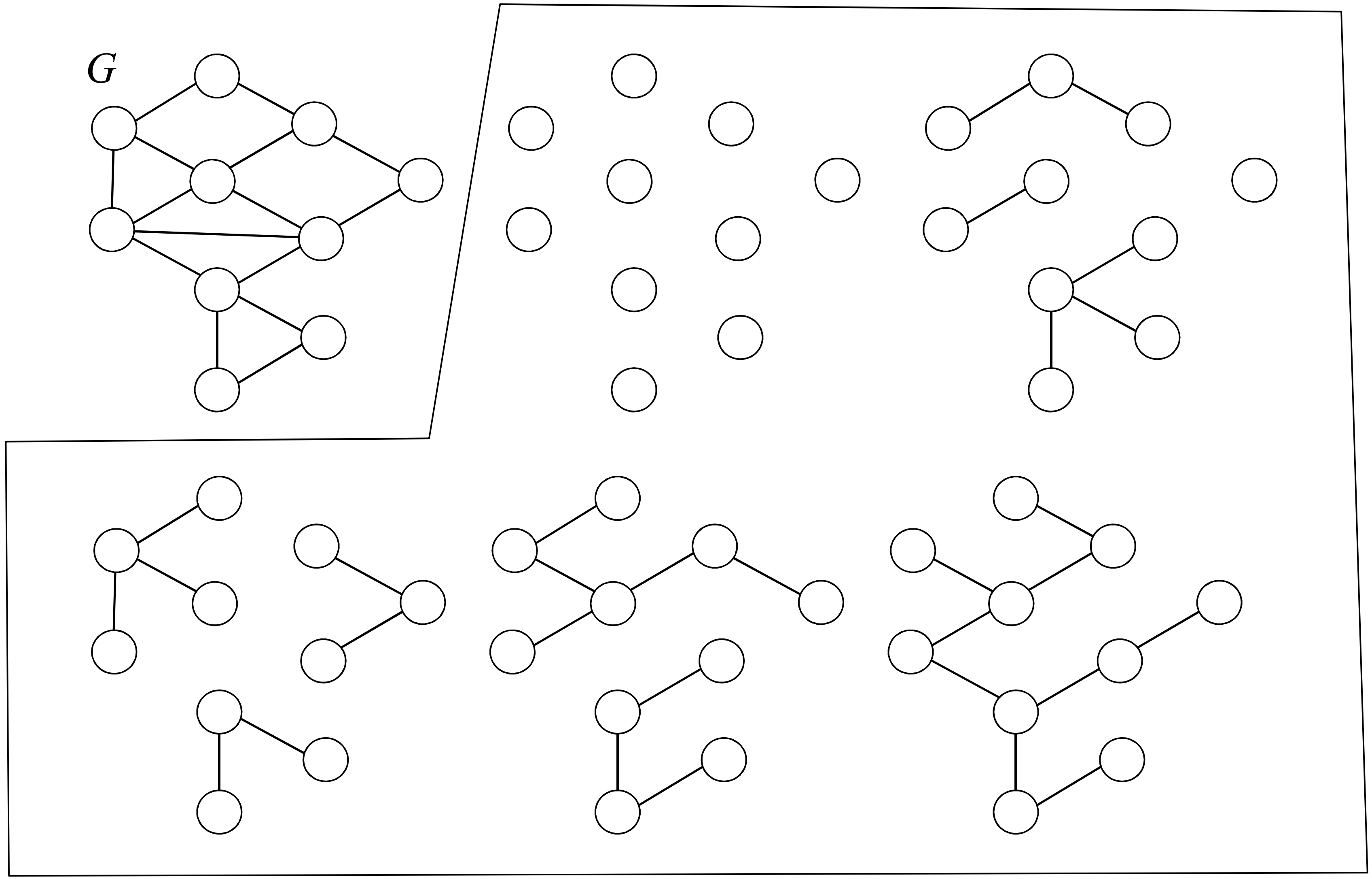
Graphic Matroid

숲forest의 집합은 매트로이드이다

- 숲forest
  - 하나 이상의 트리들로 이루어진 집합
  - 또는, 사이클을 이루지 않은 간선들의 집합

- 숲 집합 $F \subseteq 2^E$ 은 매트로이드이다

숲의 예



Weighted Matroid

- Matroid의 원집합 S의 원소들이 (양의) 가중치를 갖고 있을 때 원소들의 합을 최대화하는 부분 집합 $A \in I$를 찾고자 한다
- 아래 greedy algorithm으로 최적해가 보장된다

Greedy(I, w[])
// I : matroid, w[]: 가중치 배열
{
    $A = \emptyset$;
    S의 원소들을 의 가중치 크기로 내림차순으로 정렬한다;
    for each $x \in S$ (가중치 내림차순으로)
        if ($A \cup \{x\} \in I$) **then** $A \leftarrow A \cup \{x\}$;
    return A;
}

Weighted Matroid에 속하는 문제의 예

- Kruscal algorithm for minimum-spanning trees

Kruskal (G, r)

{

1. $T \leftarrow \Phi$; ▷ T : spanning tree
2. 단 하나의 vertex만으로 이루어진 n 개의 집합을 초기화한다;
3. Edge 집합 $Q(=E)$를 가중치가 작은 순으로 정렬한다;
4. **while** (T의 edge 수 < n-1) {

    Q에서 min-cost edge (u, v)를 제거한다;

    vertex u와 v가 서로 다른 집합에 속하면 {

        두 집합을 하나로 합친다;

        $T \leftarrow T \cup \{(u, v)\}$;

    }

}

}

(u, v)를 더함으로써 cycle을 만들지 않으면

재미있는 성질

- Weighted matroid에서 서로 다른 최적해가 2개 이상 존재하면 그들의 원소 가중치 집합은 반드시 동일하다

- Weighted matroid의 문제 공간에서는 단 하나의 봉우리만 존재하고 거기에 1개 또는 그 이상의 최적해가 존재한다 (이동 연산자와 관련이 있지만 직관적 이해를 위해 skip)

봉우리의 수

- Weighted matroid를 이루면
  - 단 1개
- TSP (w/ 2-change moving operator)
  - 10-vertex TSP – 평균 4개
  - 20-vertex TSP – 평균 170개
  - 100-vertex TSP – 평균 $3.4 * 10^{16}$개 (3경 4천조)
  - 8000-vertex TSP – ?????
- AlphaGo-Lee
  - Node 830만 개, Edge 14억 개, 서로 다른 edge 약 4백만 개
  - 봉우리가 몇 개?

봉우리의 수가 많을수록...

- 봉우리의 수가 많을수록
  - 문제 공간은 험준해진다 (rugged)
  - 풀기는 점점 어려워진다
- NP-Hard problems
  - Highly probably, 봉우리의 수 grows at least exponentially

GA가 매력적인 문제

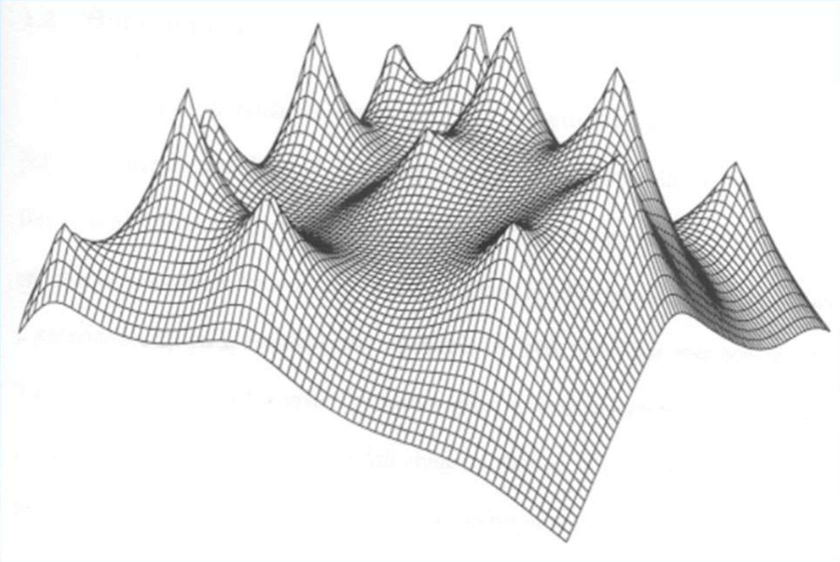
- Deterministic algorithm으로 최적해를 보장할 수 없는 문제
  - 다양한 partitioning, scheduling 문제들, NN 최적화, …
  - 대표적 class가 NP-Hard 문제들
- Deterministic algorithm으로 최적해를 구할 수 있는 문제는 GA를 쓸 필요 없다
  - 예: 최단 경로, 최소 신장 트리, …

다양한 최적화 대상들

- 카 네비게이션
- 스케쥴링
  - TSP, VRP, 작업공정, …
- Human Genome Project
  - 매칭, 계통도, functional analyses, …
- 검색
  - 데이터베이스, 웹페이지들, …
- 자원의 배치
- 반도체 설계
  - Partitioning, placement, routing, …
- …

✓ 이들 중 어려운 문제만 GA에게 매력이 있다

공간의 여행

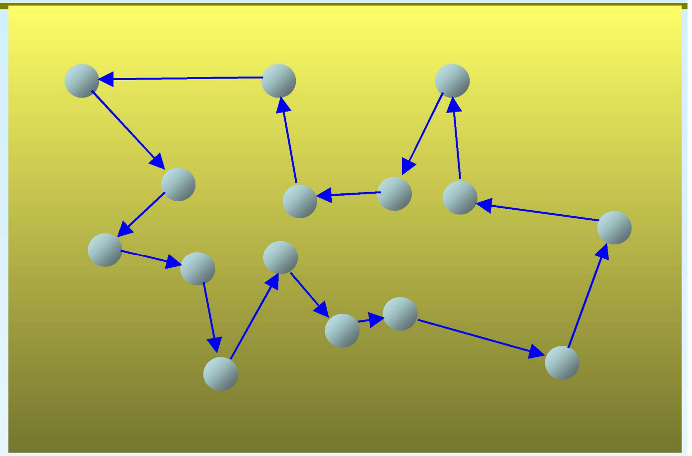


문제 = 공간

알고리즘 = 여행수단




도대체 얼마나 복잡하길래: TSP



TSP

N개 지점을 다 방문하고 원점으로 돌아오는 최단경로는?

질문:

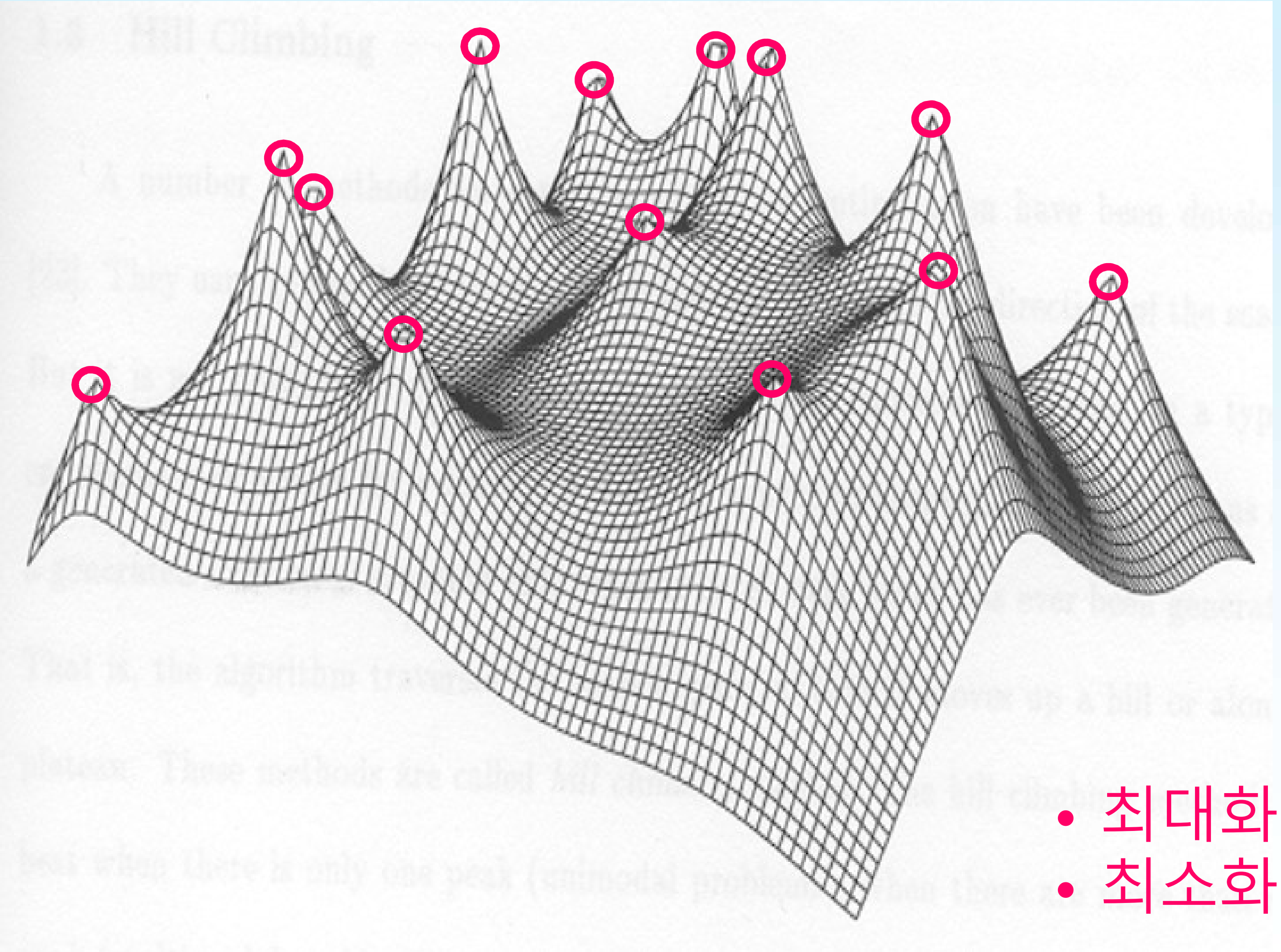
(컴퓨터가 1초에 150만 경우의 수를 평가한다면) 27개 지점을 다 방문하고 돌아오는 모든 경우(26!)를 보는데 드는 시간?

1분 1시간 1일 1달 1년 1000년 10억년 1조년

10조년

✓ 요즘은 십만개 짜리 문제를 다룬다!!

끌개 - **Local Optimum**

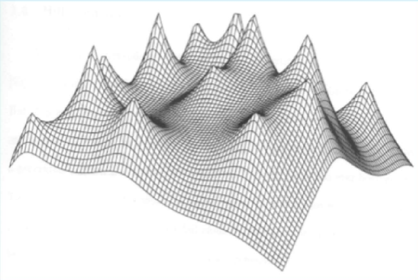


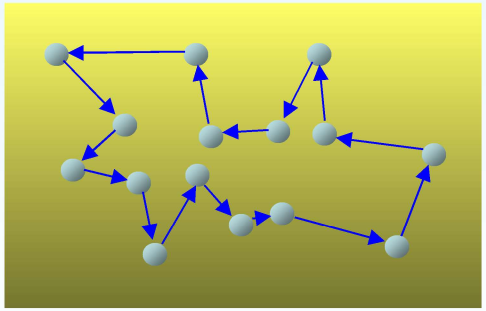
○: **지역최적점**
(local optimum)
=
끌개
(attractor)

- 최대화 문제 : 봉우리
- 최소화 문제 : 계곡

방대하고 황량한 공간: TSP의 끌개수

문제 크기(지점수)	끌개의 수(평균)
10	4
20	170
100	3.4×10^{16} (3경 4천조)



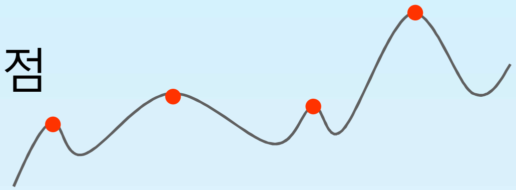


모든 솔루션의 수: 9.3×10^{157}
끌개의 비율: 3×10^{141} 솔루션당 하나씩의 끌개

- ✓ 지금은 8천개짜리 문제의 최적해를 구한다
- ✓ 최적화 알고리즘은 이런 공간을 돌아다니는 교통수단이다

끌개 (Attractor)

Attractor = 끌개
- 문제공간 상에서의 국소적 최적점
- 공간탐색의 목표이자 장애물



끌개의 예
- 생태계의 종: 개나리, 질경이, 치타, 가젤, ...
- 인류사의 조직, 제도: 가족, 부족, 국가, 학교, 대통령제, ...
- 인간의 고정 관념, 사고 체계: 시각, 습관, 편집증, ...
- 시장에서 정착되는 제품들
- 알고리즘이 만들어내는 주식투자전략
- 알파고의 가치망
- 테니스의 스윙폼
- ...

Optimization은 가장 수준 높은 끌개를 찾는 것
- 저수준 끌개(low-quality local optimum)에 고착되어 버리지 않도록
- 다양한 끌개에 접할 수 있도록 넓은 탐색 기능 필요

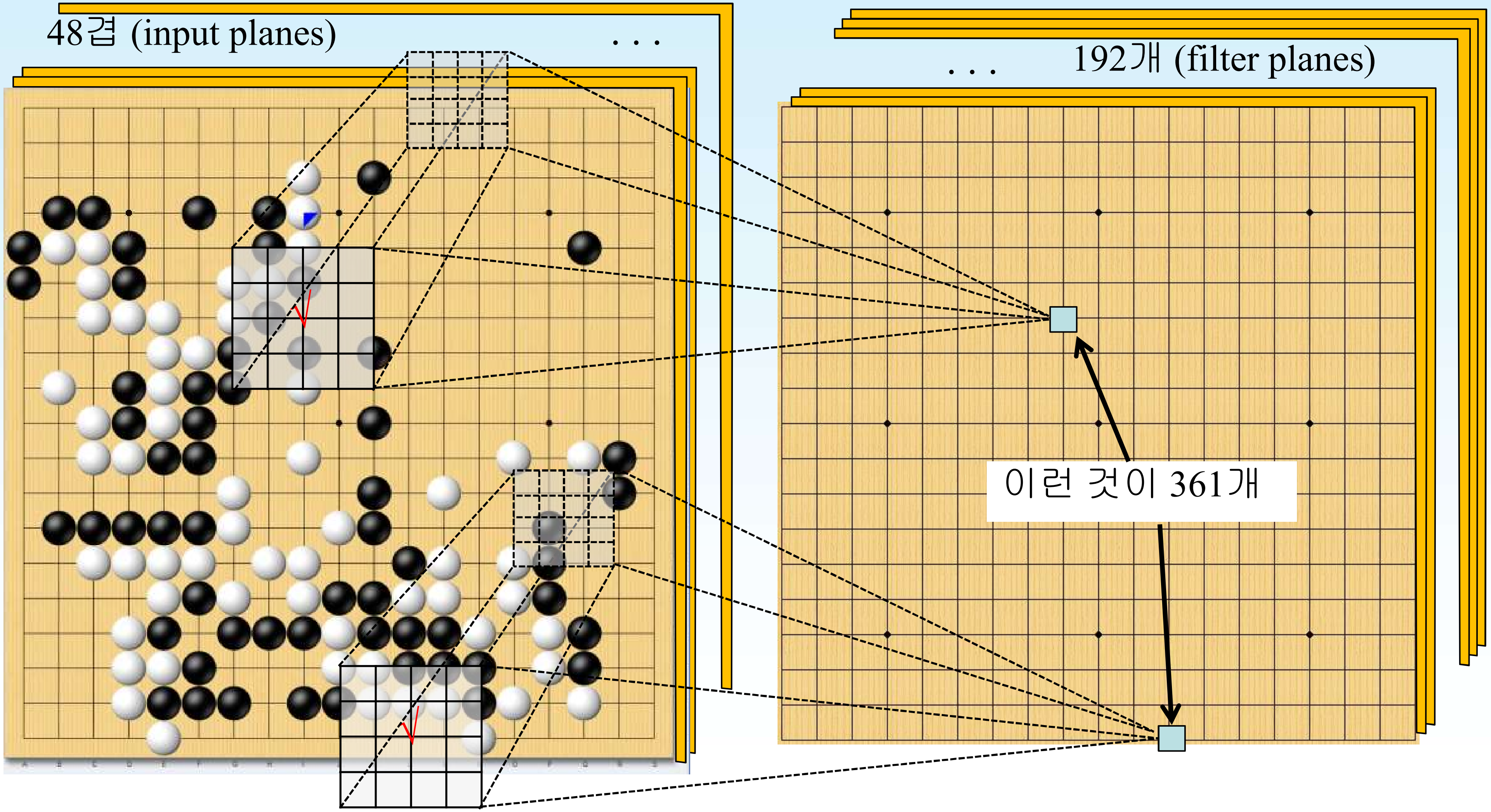
공간탐색에서의 최대 장애물 – 끌개

- Attractor에서 벗어나는 데는 상당한 에너지가 든다
  - 굳어진 스윙폼 교정, 고정관념의 변경, 제도 개혁
  - Large-step Markov chain
    - perturbation + local optimization
  - Genetic algorithm
    - Crossover, mutation
- Revolution

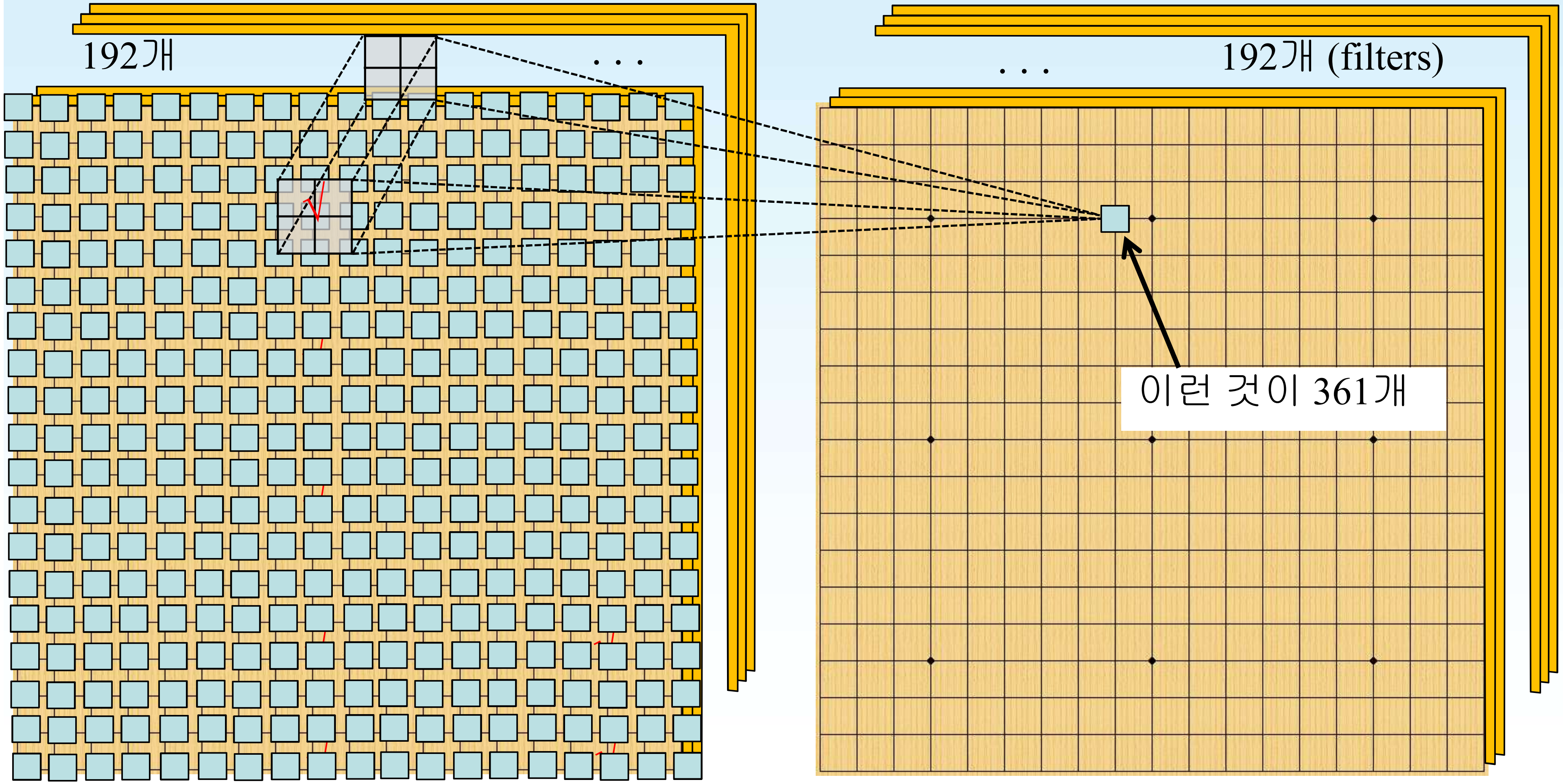
  = Strong perturbation + local optimization

✓ 끌개는 공간탐색의 목표이자 장애물이기도 하다.

알파고가 본 바둑, AlphaGo-Lee: CNN



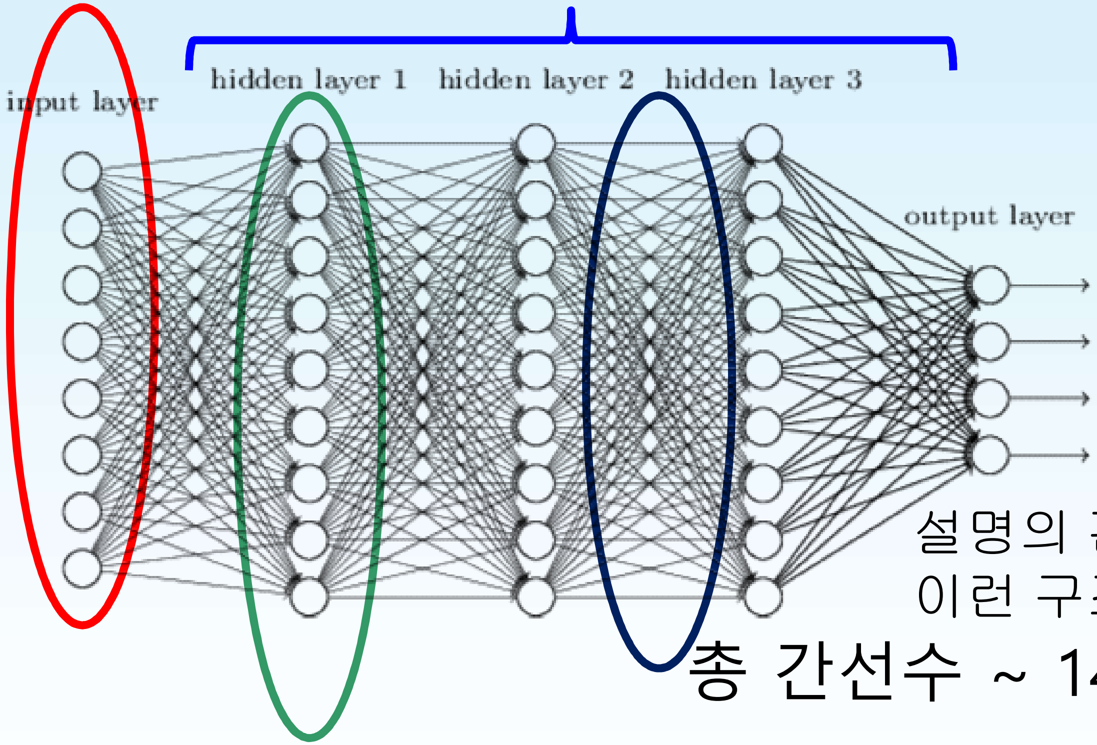
이런 식으로 11번 반복 (12층)



놀라운 규모: 알파고

입력 단자 ~ 25,000개

Layer 수 = 12개, 13개



설명의 편의상...
이런 구조는 아니다

총 간선수 ~ 14억개

Hidden Node 수 (중간 마디수) ~ 830만개

✓4백만 차원 vector의 optimal assignment 문제

동물의 뉴런 수

	#neurons	#synapses	#neurons in cortex
꿀벌	96만	10억	
정책망(알파고)	**830만**	**14억**	
쥐	7000만	1000억	
고양이	7억6천	10조	
침팬지	70억		
인간	**860억**	**100조~1000조**	200억
아프리카 코끼리	2300억		110억
참거두(돌)고래			370억

몬테카를로 롤아웃 (Monte Carlo Rollout)

- 바둑판의 어떤 상태에서 게임이 끝날 때까지 가본다
  - AlphaGo-Lee: heuristic moves
  - AlphaGo-Zero: NN 자신을 사용, 강화 학습
- 수천~1만번 rollout 후 승률 계산