SOON CHUN HYANG UNIVERSITY

2024 Probabilistic Model Class

Modern Optimization with R

- Chapter 3. Blind Search

SAIL
Senseable Artificial Intelligence Laboratory

순천향대학교 미래융합기술학과

Senseable AI Lab

석사과정 김병훈

Index
Contents

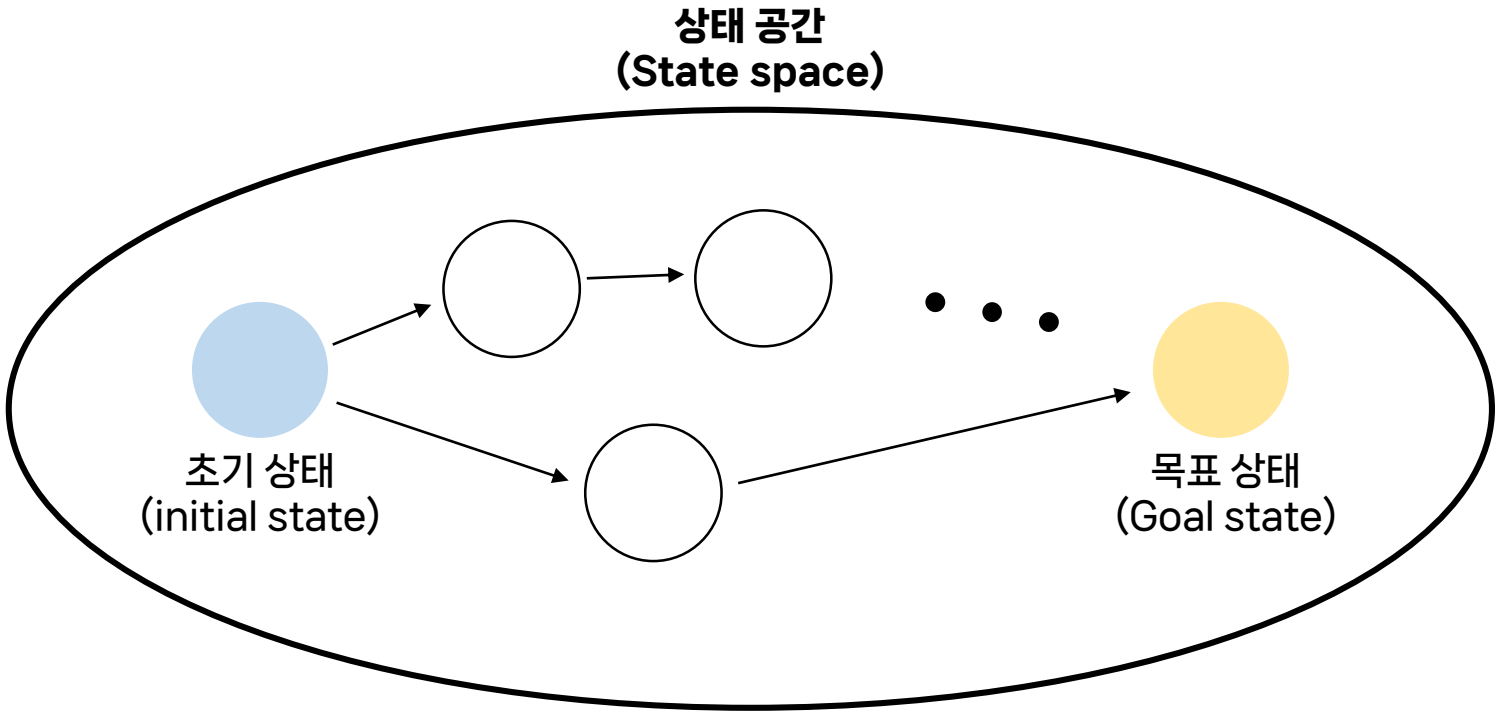
1 Introduction

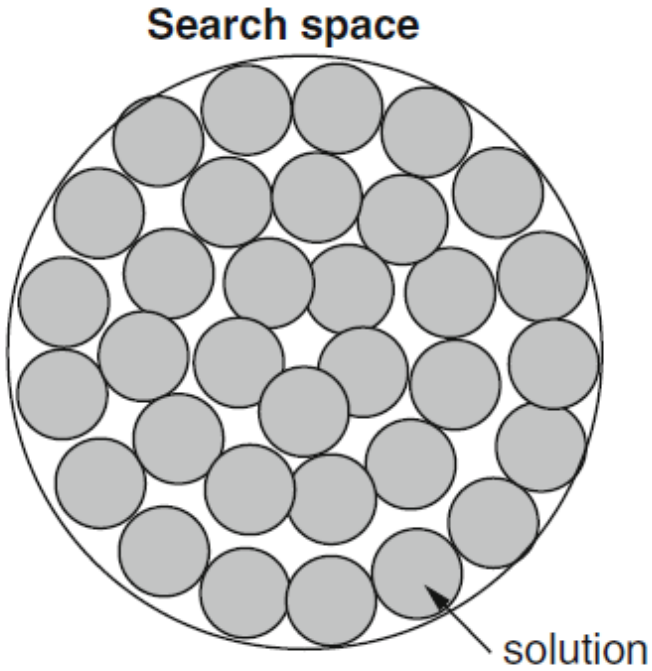
1. Introduction

Search Algorithm

탐색(Search)이란?

- 정의 : 문제의 해(Solution)가 될 수 있는 것들의 집합을 공간(Space)로 간주하고, 문제에 대한 최적의 해를 찾기 위해 공간을 체계적으로 찾아보는 것
- 필요성 : 많은 시간 소요, 메모리 문제





1. Introduction

Search Algorithm

Search Algorithm의 종류

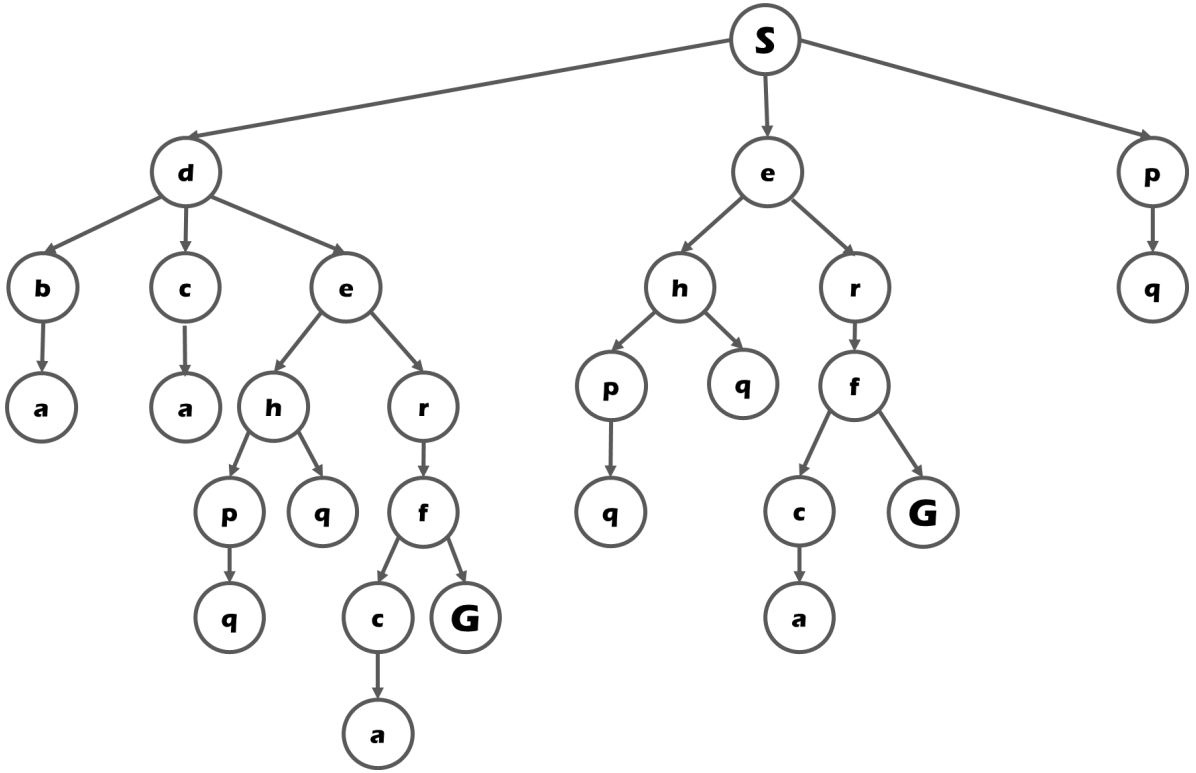
- Blind Search(Uninformed search)
- Heuristic Search(Informed search)
- Local Search
- Adversarial Search

1. Introduction

Blind Search Algorithm

Blind Search Algorithm

- 문제에 대한 정의 외에 다른 어떠한 정보가 주어지지 않음 -> 모험을 하듯이 탐색하는 기법!
- Main Question : node 중 어느 것을 골라야, Goal으로 연결되는지 테스트 하는 전략

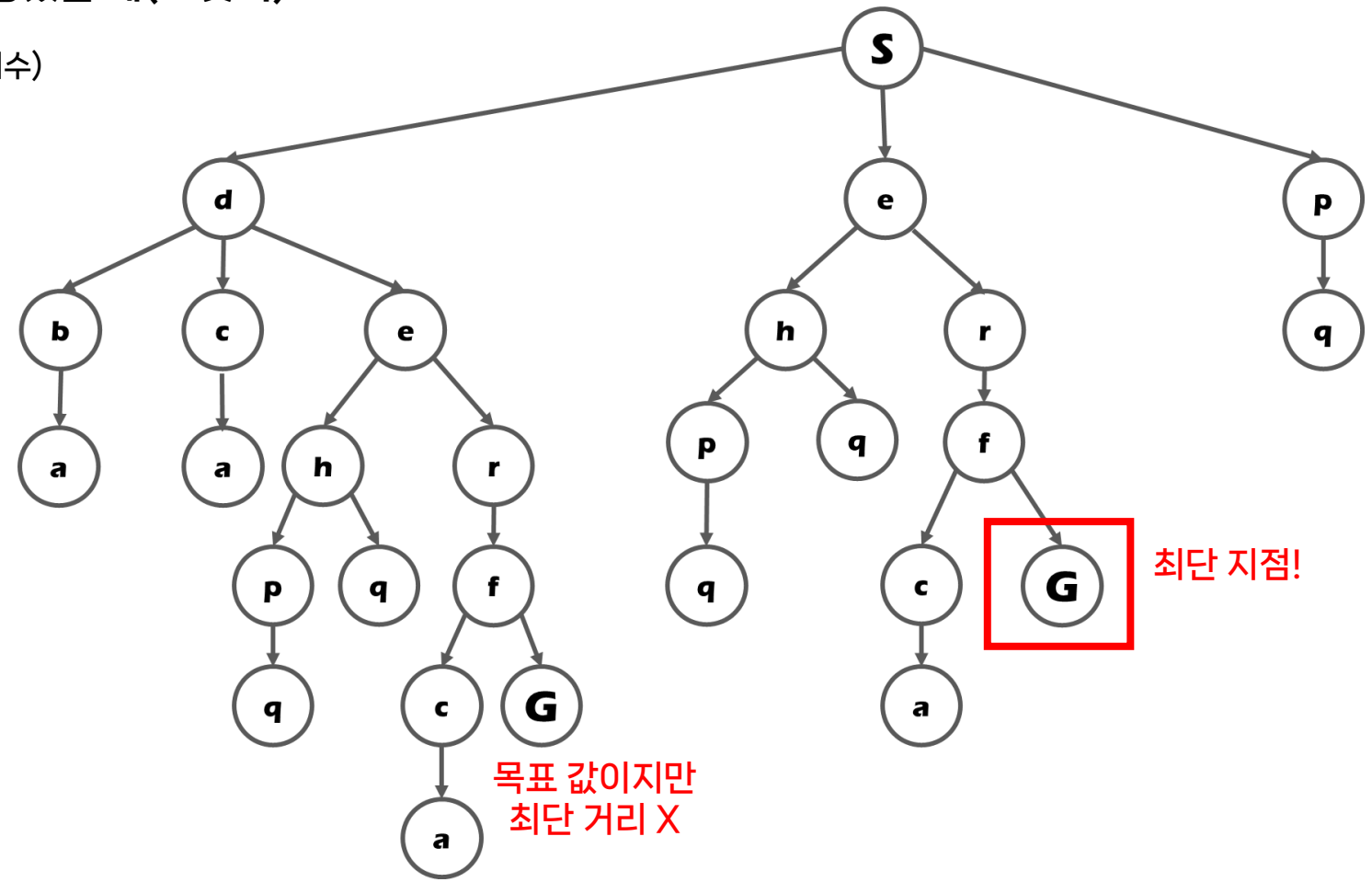


1. Introduction

Blind Search Algorithm

각 노드 간의 거리 비용을 1로 가정했을 때(G 찾기)

- 시간 복잡도 $O(N^2)$ (N : 정점 개수)

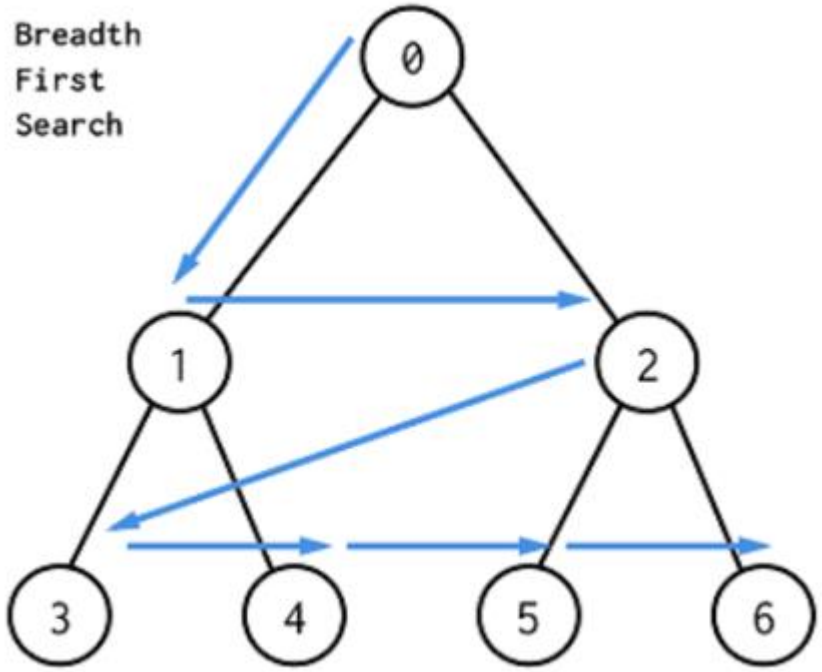


1. Introduction

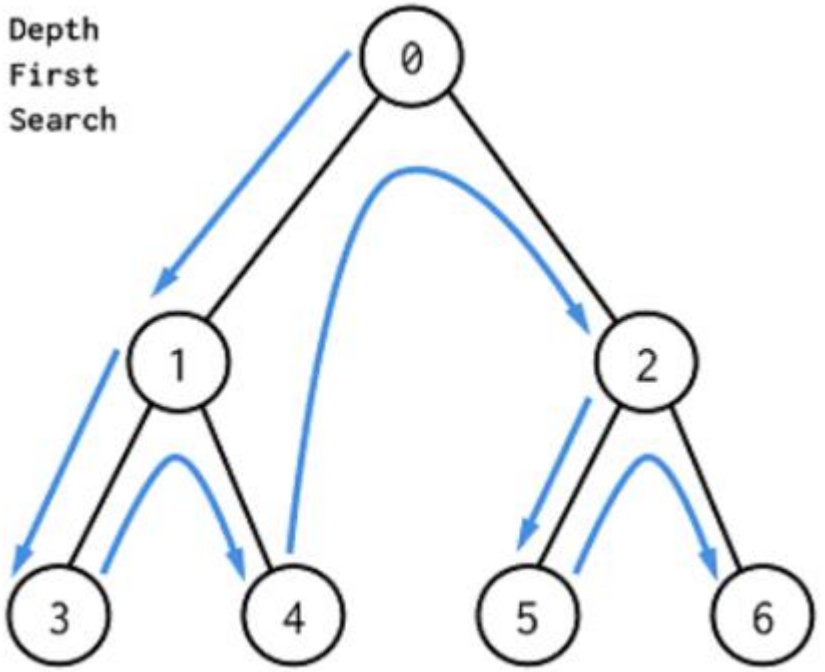
Blind Search Algorithm

Types of Blind Search Algorithm

- BFS(너비우선탐색)
- DFS(깊이우선탐색)



Breadth First Search



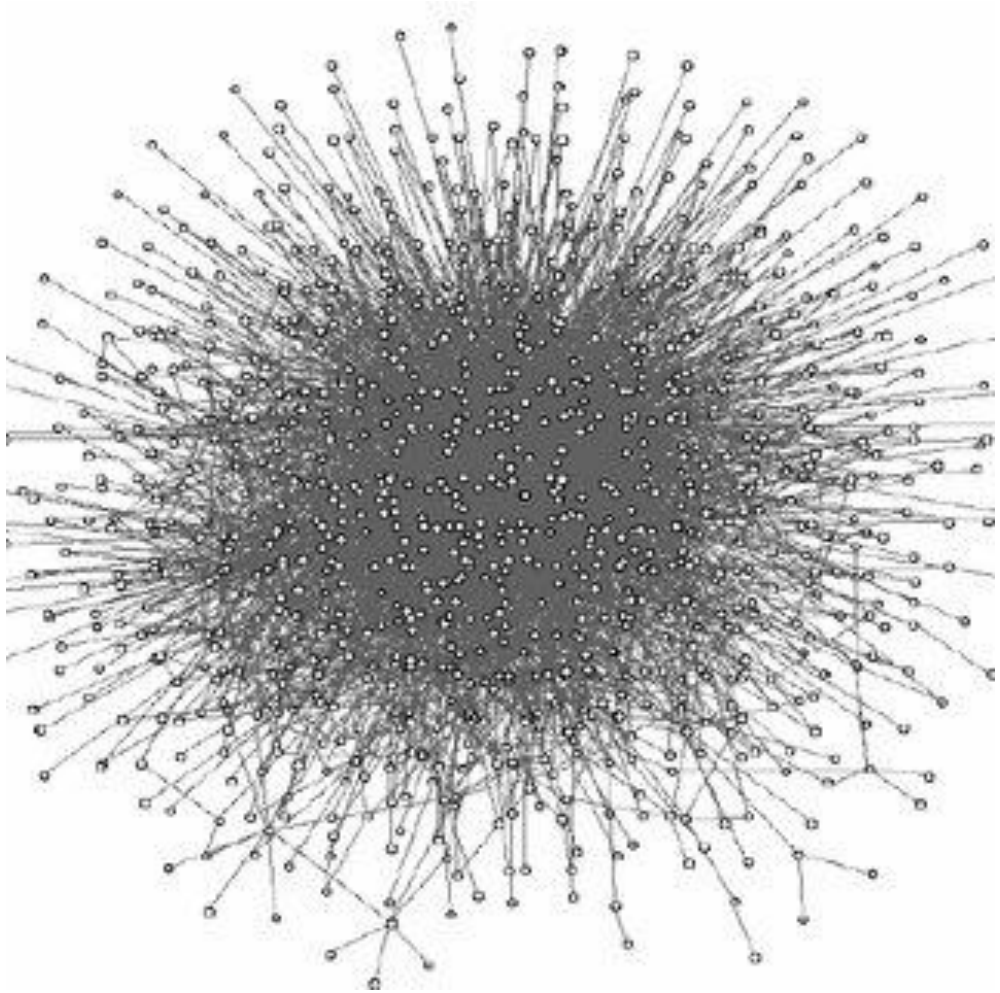
Depth First Search

1. Introduction

Blind Search Algorithm

Limitation of Blind Search Algorithm

- 맹목적으로 목표를 찾기 때문에 비효율적
- 목표가 어디에 있는지에 따라 극단적인 상황이 나올 수 있음



실상황(P2P 네트워크)에서의 그래프 표현

1. Introduction

Blind Search Algorithm

Bag Price : 가방 제조 공장에서 생산된 가방의 판매 가격 설정 문제(정수 최적화 문제)

제조 비용 : $30
마케팅 비용 : $2

제조 비용 : $25
마케팅 비용 : $1.75

제조 비용 : $20
마케팅 비용 : $1.5

제조 비용 : $15
마케팅 비용 : $1.25

제조 비용 : $10
마케팅 비용 : $1

$$cost = 100 + Production\ cost \times sales$$

$$sales = (1000/\ln(x + 200) - 141) \times Marketing\ cost$$

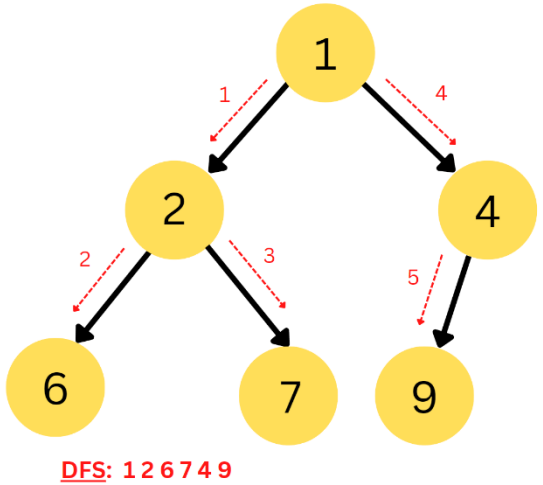
판매 가격 x가 $1 ~ $1000의 값을 가진다면, 검색 횟수 = 1000^5

1. Introduction

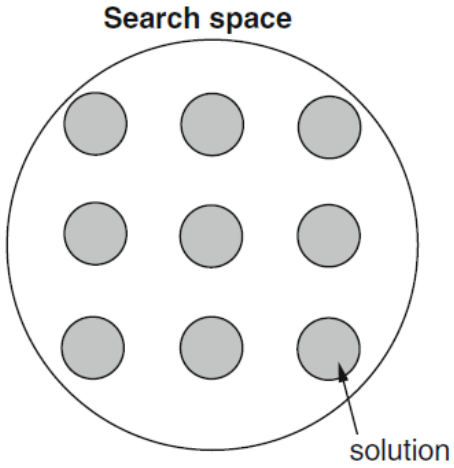
Blind Search Algorithm

Solution

- 따라서, 임계값을 설정하거나(Depth-Limited Search)
- 검색 공간을 줄이거나(Grad Search, Monte Carlo Search)
- 휴리스틱 기법을 사용하거나



Depth-Limited Search



Grad Search

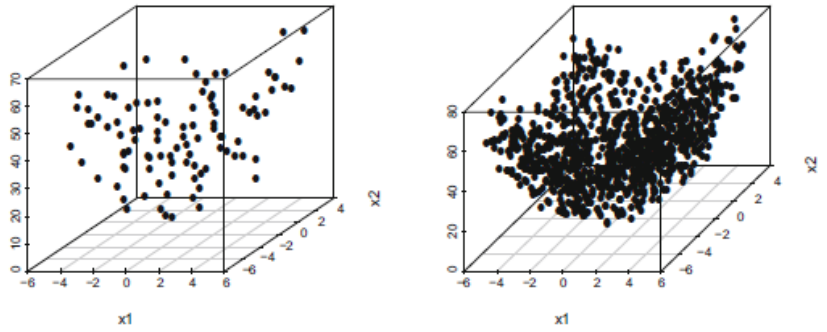


Fig. 3.3 Example of Monte Carlo search using $N = 100$ (left) and $N = 1000$ (right) samples for sphere and $D = 2$

Monte Carlo Search

2 Full Blind Search

2. Full Blind Search

Define

Problem

1. 이진법 사용
    1. 비트 합 최적화
    2. 최대 사인 해 최적화
2. Bag Price 문제

Function

1. fsearch : 전체 검색 공간을 행렬로 정의, 각 행을 순차적으로 평가 -> 단순한 Blind Search Algorithm
2. dfsearch : 재귀적 방법을 사용하여 깊이 우선 탐색을 수행 -> 좀 더 복잡한 Blind Search Algorithm

2. Full Blind Search

이진법(비트 합 최적화)

Pseudo code

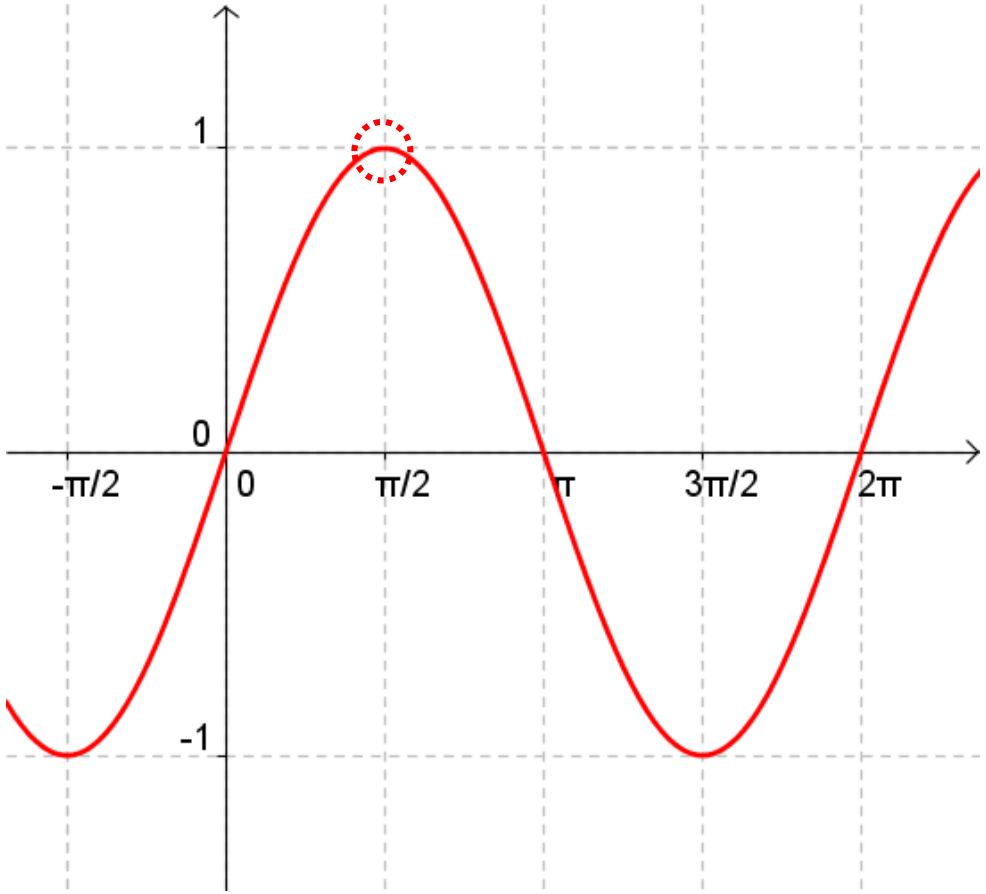
```
function fsearch(search, fn, type)
    Set bestValue to 0
    For each solution in search
        Calculate value using fn(solution)
        If value > bestValue then
            Update bestValue and bestSolution
    Return bestSolution and bestValue
end function
```

```
function dfsearch(level, domain, fn, type, D, parameters, bestCurrent)
    If level > D then
        currentValue := fn(parameters)
        If currentValue > bestCurrent.value then
            bestCurrent.value := currentValue
            bestCurrent.solution := parameters
        Return bestCurrent
    Else
        For each value in domain[level]
            parameters[level] := value
            bestCurrent := dfsearch(level + 1, domain, fn, type, D, parameters, bestCurrent)
        Return bestCurrent
end function
```

```
fsearch best s: 1 1 1 1 1 1 1 1 f: 8
dfsearch best s: 1 1 1 1 1 1 1 1 f: 8
```

2. Full Blind Search

이진법(최대 사인 해 최적화)




```
sin(pi*(intbin(x))/(2^Dim)) == sin(pi*(1/2))
```

$$\sin\frac{intbin(x)}{2^D}\pi = \sin\frac{1}{2}\pi$$

$$\frac{intbin(x)}{2^8} = \frac{1}{2}$$

$$intbin(x) = 128$$

```
fsearch best s: 1 0 0 0 0 0 0 0 f: 1
dfsearch best s: 1 0 0 0 0 0 0 0 f: 1
```

2. Full Blind Search

Bag Price 문제

Result

- fsearch 함수를 이용하여 최적화 함
- 하지만, 1000^5로 설정하면 불가능하므로, 각 가방은 독립적으로 존재한다고 가정하여 1000×5 로 계산

```
optimum s: 414 404 408 413 395 f: 43899
```

각 가방이 위의 값일 때, 가방 공장의 이윤 최대화

3 Grid Search

3. Grid Search

Defined

Grid Search란?

정의 : 주어진 검색 공간 내에서 균일하게 분포된 Grid 상의 점들을 평가(차원 축소)

주요 절차: 검색 공간 설정 -> Grid Step 설정 -> Grid Point 평가 -> 최적 해 탐색

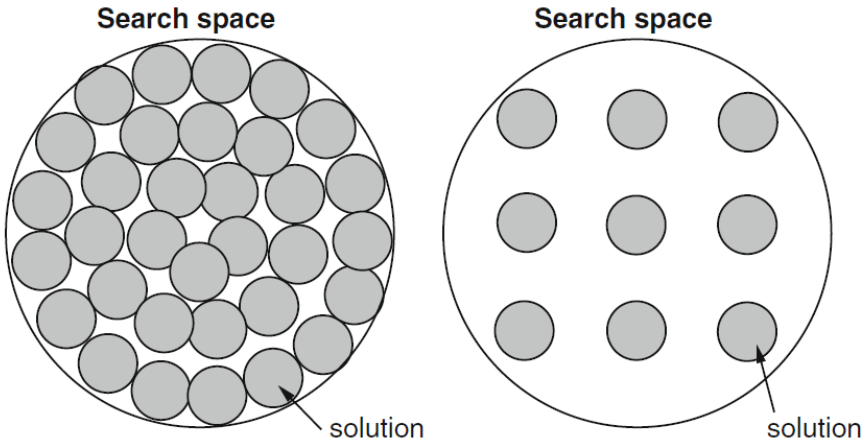


Fig. 3.1 Example of pure blind search (left) and grid search (right) strategies

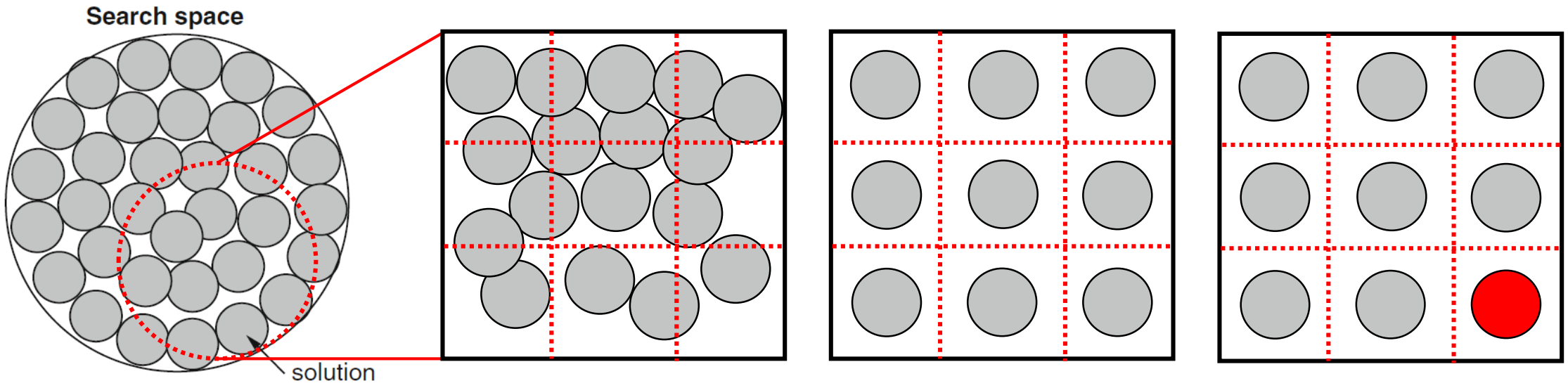
Function

1. Standard Grid Search : 일반적인 Grid Search
2. Uniform Design Search : 더 적은 Point 사용
3. Nested Grid Search : 점차 작은 Step을 사용해서 더 최적의 해 탐색(이전 단계의 결과에 기반하기 때문에 탐욕적 휴리스틱 방법)

3. Grid Search

Standard Grid Search

검색 공간 설정 -> Grid Step 설정 -> Grid Point 평가 -> 최적 해 탐색

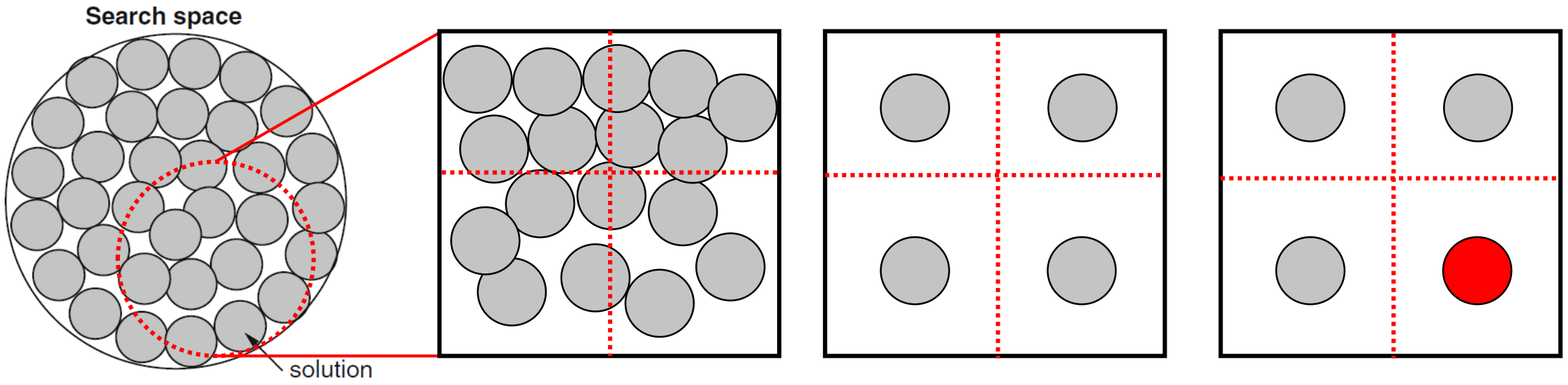


3. Grid Search

Uniform Design Search

검색 공간 설정 -> Grid Step 설정 -> Grid Point 평가 -> 최적 해 탐색

- 더 작은 검색 공간(균일 분포)

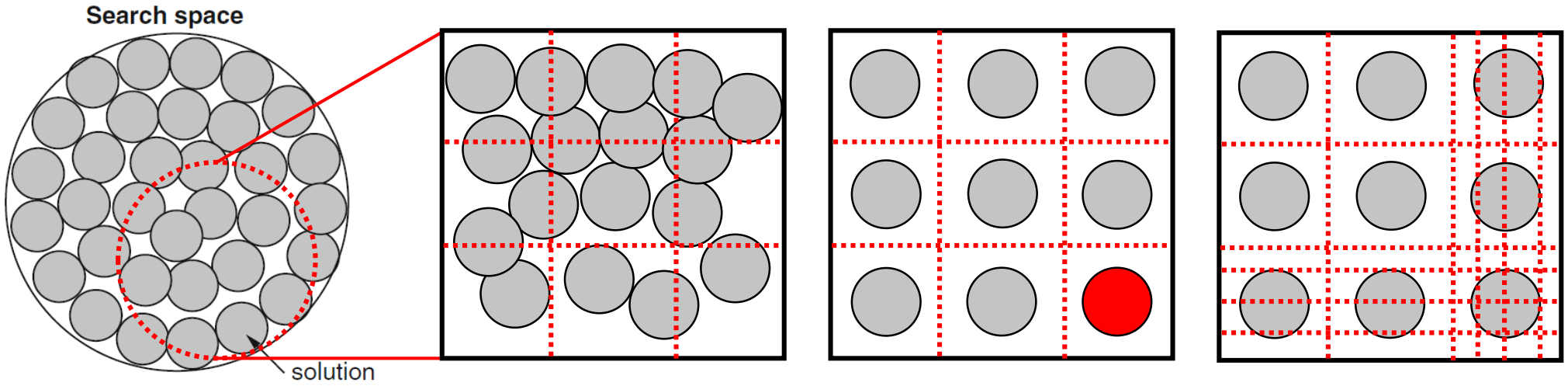


3. Grid Search

Nested Grid Search

검색 공간 설정 -> Grid Step 설정 -> Grid Point 평가 -> 최적 해 탐색

- 점차 작은 Step을 사용



3. Grid Search

Limitation

Limitation

- 차원의 저주 : 변수의 수가 증가함에 따라, 필요한 계산량이 기하급수적으로 증가
- 정밀도와 계산 비용의 반비례 : 그리드 수를 증가시키면 정밀도는 향상되지만, 계산 비용 또한 증가
- 지역 최소값(local minima) : 지역 최소값에 갇힐 수 있음 -> 최적의 해를 찾지 못하는 문제 발생
- 파라미터 설정 : 추가적인 파라미터를 직접 설정해야 함 -> 최적의 파라미터를 또 찾아야함..

여전히 시간 복잡도 $\boldsymbol{O(N^2)}$

4 Monte Carlo Search

4. Monte Carlo Search

Defined

Monte Carlo Search란?

정의 : 주어진 확률 분포를 사용하여 무작위로 N개의 점을 생성하여 가장 목표값과 근사한 값을 찾는 방법

※ 해당 책에서는 균일 분포 사용

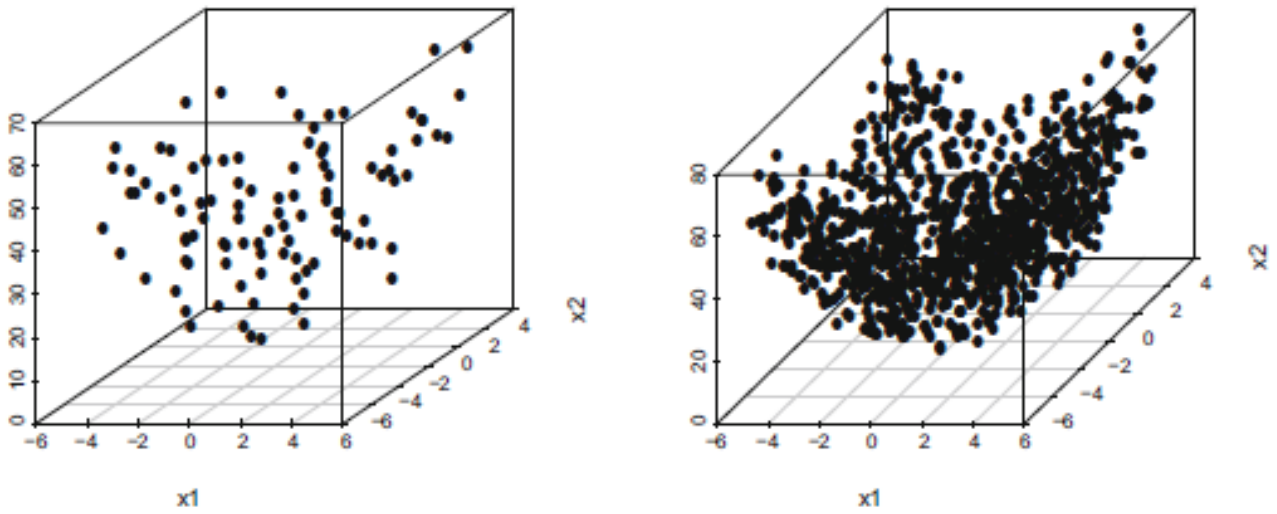


Fig. 3.3 Example of Monte Carlo search using $N = 100$ (left) and $N = 1000$ (right) samples for sphere and $D = 2$

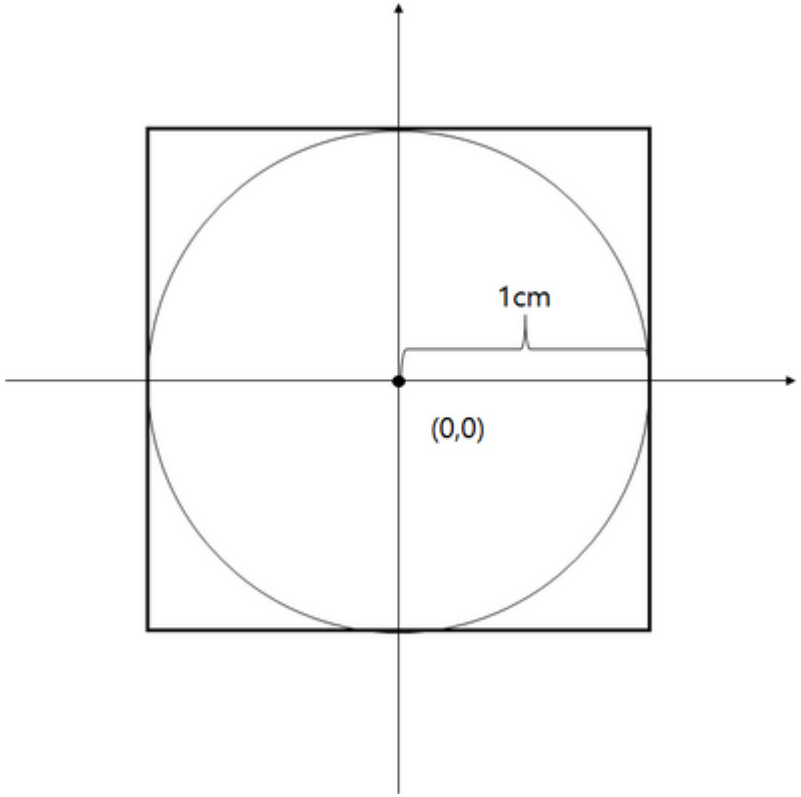
4. Monte Carlo Search

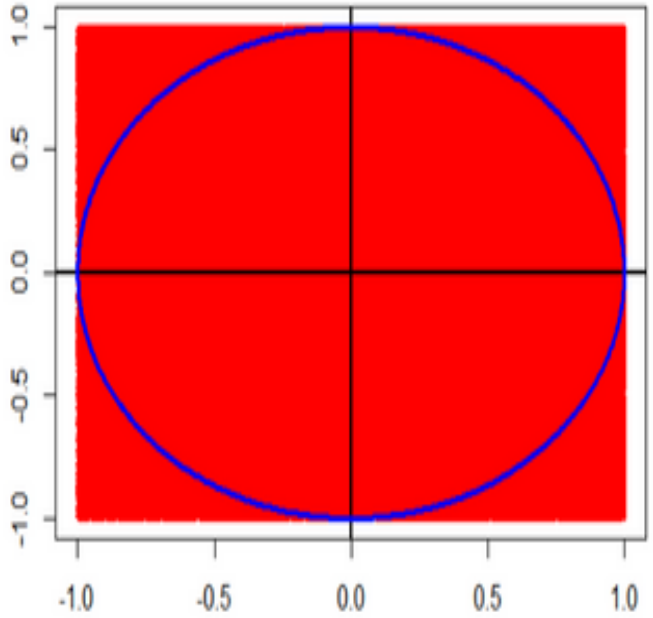
Example

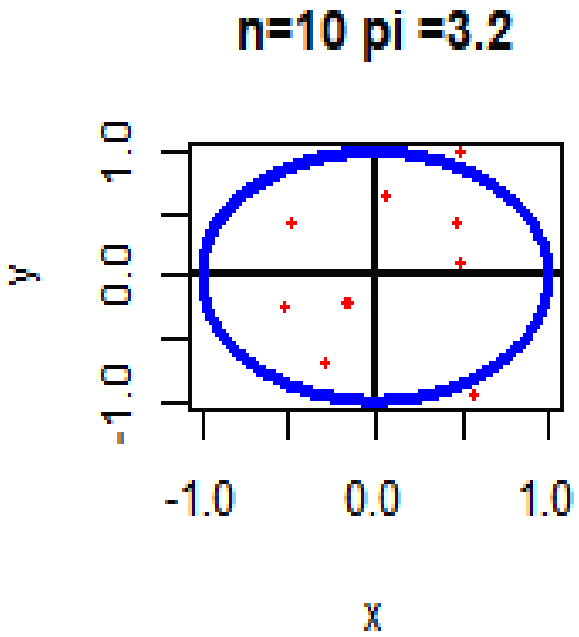
Example of Monte Carlo Search

주어진 사각형에 내접하는 원이 존재할 때, 100만개의 점을 찍고 원 안에 존재하는 점의 개수를 세면?

-> 원주율에 근사한 값을 구할 수 있음







4. Monte Carlo Search

vs. Grid Search

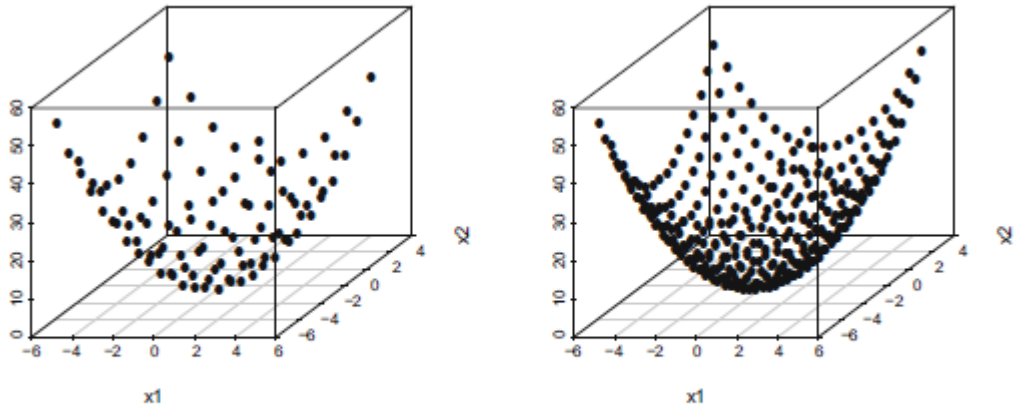


Fig. 3.2 Example of grid search using $L = 10$ (left) and $L = 20$ (right) levels for **sphere** and $D = 2$

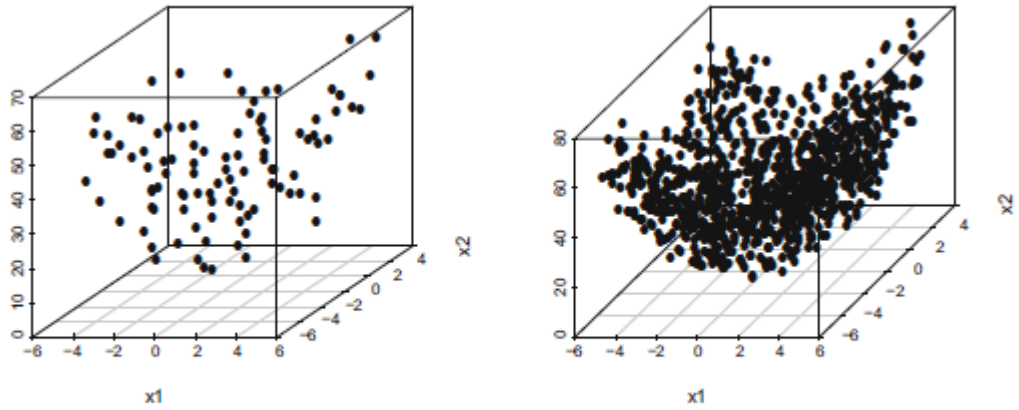


Fig. 3.3 Example of Monte Carlo search using $N = 100$ (left) and $N = 1000$ (right) samples for **sphere** and $D = 2$

```
sphere:
gsearch s: 0.3 0.3 f: 0.18
ngsearch s: -0.1 -0.1 f: 0.02
rastrigin:
gsearch s: -1.9 -1.9 f: 11.03966
ngsearch s: -0.1 -0.1 f: 3.83966
```

```
sphere D: 2 s: -0.01755296 0.0350427 f: 0.001536097
sphere D: 30 s: -0.09818928 -1.883463 f: 113.7578
rastrigin D: 2 s: -0.0124561 0.02947438 f: 0.2026272
rastrigin D: 30 s: 0.6508581 -3.043595 f: 347.1969
```

4. Monte Carlo Search

Limitation

Limitation

- 일관성이 떨어짐
- 차원이 증가하면 수렴 속도가 낮아짐
- 샘플 N의 개수가 많을수록 근사치에 가까워지지만, 시간 복잡도가 올라감
- 비효율적인 샘플링 방법으로 인해 중요하지 않은 영역에 대한 계산과정이 일어남

But, 시간 복잡도가 Full Blind Search, Grid Search와 달리 $O(N)$으로 크게 줄어듦

Source Code

Github : https://github.com/Kim-Byeong-Hun/Probability_Modeling

Thanks