Traditional Methods – Part 1

Exhaustive Search

- Exhaustive search
  - search space 내의 모든 가능한 solution 탐색
  - global solution 가능, but search space 가 작은 경우에만 사용 가능
  - 알고리즘
    - 단순
    - 어떻게 가능한 solution 들을 순차적으로 발생시키는 가?

Exhaustive Search

- SAT

: n-bits binary string을 어떻게 생성할 것인가?

<방법 1>

- 0 ~2^{n-1} 의 정수 생성
- 각 정수를 bit 의 2 진수로 변환

(ex) 0 → 0000, 1 → 0001, 2 → 0010, …… , 15 → 1111

<방법 2>

- (0 0 … 0)에서 시작
- (0 0 … 1)을 더하여 새로운 string 생성

Exhaustive Search

- SAT

<방법 3> depth-first search

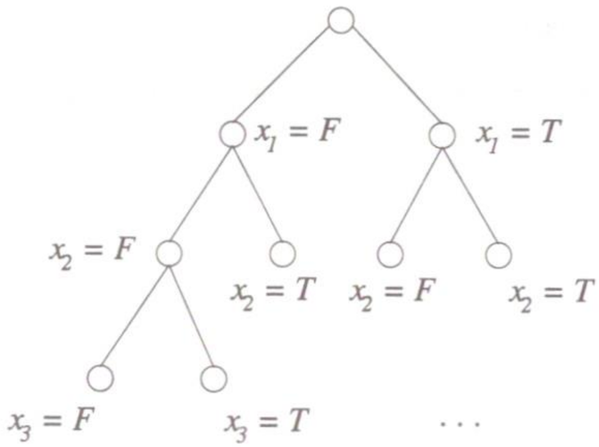


Fig. 3.1. A binary search tree for the SAT

```
procedure depth-first(v)
begin
    visit v
    for each child w of v do
        dept-first(w)
end
```

Fig. 3.2. Depth-first search

search space 중 일부를 제거하고 탐색 진행 가능

(ex) $\cdots \wedge (x_1 \vee \overline{x_3} \vee x_4) \wedge \cdots$

$x_1 = F, x_2 = T, x_3 = T, x_4 = F$

or

$x_1 = F, x_2 = F, x_3 = T, x_4 = F$

=> 하위 노드에 대한 탐색 불필요

Exhaustive Search

- TSP

  n 개의 자연수 (도시) 에 대한 순열 (permutation, 방문순서)를 어떻게 생성할 것인가?

  (ex) n = 3

  (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2) (3,2,1)

```
procedure gen1_permutation(i)
begin
  k ← k + 1
  P[i] ← k
  if k = n then
     for q = 1 to n do
        print P[q]
  for q = 1 to n do
     if P[q] = 0 then gen1_permutation(q)
  k ← k − 1
  P[i] ← 0
end
```

Exhaustive Search

- NLP

  : 실수 $x_1, \cdots, x_n$에 대한 탐색 cell 생성

  (ex) $-1 \leq x_1 \leq 3, \quad 0 \leq x_2 \leq 12,$

  $\Delta x_1 = 0.01$, $\Delta x_2 = 0.02$,

  => x_1: 400 intervals

  x_2: 600 intervals

  => total 400 x 600 = 240,000 cells

Local Search

- Search space
  - exhaustive search: entire space
  - local search: local neighborhood
- Procedure

  S1. Pick a solution from the search space and evaluate its merit. Define this as the *current* solution

  S2. Apply a *transformation* to the current solution to generate a *new* solution and evaluate its merit.

  S3. If the new solution is better than the current solution then exchange it with the current solution; otherwise discard the new solution.

  S4. Repeat S2 and S3 until no transformation in the given set improves the current solution.

※ 'transformation'을 어떻게 정의할 것인가?

※ 'initial solution'을 어떻게 결정할 것인가?

Local Search

- SAT
  - Neighborhood: 1-flip mapping

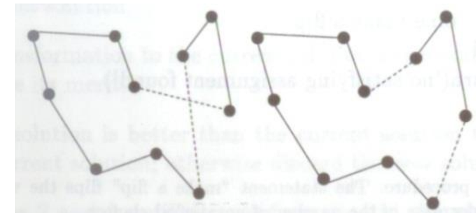
```
procedure GSAT
begin
    for i ← 1 step 1 until MAX-TRIES do
    begin
        T ← a randomly generated truth assignment
        for j ← 1 step 1 until MAX-FLIPS do
            if T satisfies the formula then return(T)
            else make a flip
    end
    return('no satisfying assignment found')
end
```

Local Search

- TSP

  <방법> 2-opt

  Neighborhood: 2-swap mapping



```
procedure 2-opt
begin
    for i←1 step 1 until MAX_TRIES do
    begin
        generate a tour T
        BestCost = Cost(T)
        for j=1 step 1 until n−1 do
        begin
            Improve=FALSE
            for k=j+1 step 1 until n do
            begin
                generate a tour T' by swap(j,k)
                if Cost(T') < BestTour then
                    BestTour = Cost(T')
                    best_k = k
                    Improve = TRUE
                Restore tour T by swap(j,k)
            end
            if(Improve = TRUE)
                generate a new tour by swap(j,best_k)
        end
    end
end
```

☞ k-opt

Local Search

- NLP

  <방법> Gradient Method of minimization

  min $f(\boldsymbol{x})$

  - Update rule: steepest decent

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

$$\nabla f(\boldsymbol{x}_k) = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}]_x$$

  - Newton's method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (H(f(\boldsymbol{x}_k)))^{-1} \nabla f(\boldsymbol{x}_k)$$

$$H(f(\boldsymbol{x}_k)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$: Hessian matrix

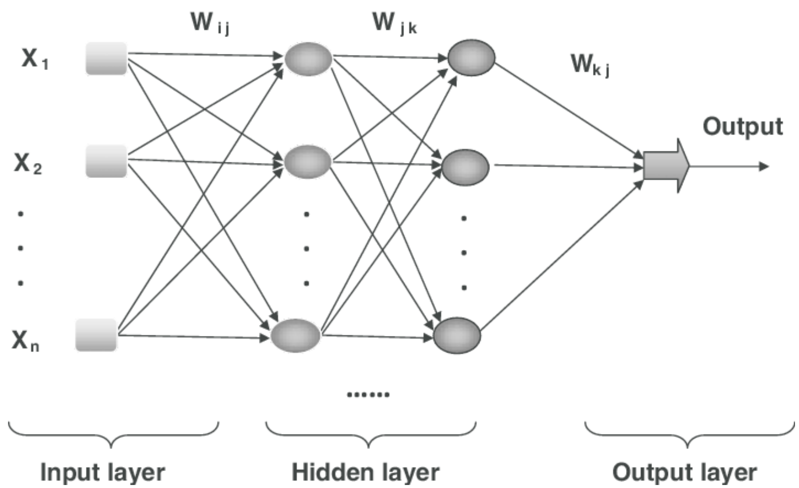
Local Search

- NLP

(ex) $\min f(x_1, x_2) = x_1^2 + x_2^2$

Local Search

Ex) Error Backpropagation Algorithm in Neural Network



Weight Update Rule

Old weight

Derivative of Error with respect to weight

$$^*W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

New weight

Learning rate

Gradient decent