TOPIC NAME : Lecture-6 DAY : TIME : DATE : / /

Local Search Algorithms

Optimization search problems :

- Different from path finding search-
  - No fixed start state
  - The path to the solution is not important.

- Focus
  - Define an objective function that measures the quality of a solution
  - Goal is to maximize or minimize this function.

The Traveling Salesman problem

The TSP states that we're Salesperson and we must visit a number of cities or towns.

Rules : visit every city only once, then

GOOD LUCK

TOPIC NAME : DAY :
TIME : DATE : / /

return back to the city we started in

Goal : Find the shortest possible route

How it works :

1. Check the length of every possible route, one route at a time
2. Is the current route shorter than the shortest route found so far? If so, store the new shortest route
3. After checking all routes, the stored route is the shortest one.

Such a way to finding the solution to a problem is called brute force

GOOD LUCK™

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Approaches for solving

1. Brute force search (Exhaustive)
   - Try all possible routes
   - Complexity O(n!) [factorial]
2. Dynamic Programing (Held karp algo)
   - Uses subsets & recursion
   - Complexity [$O(n^2 2^n)$] better than factorial

Branch & Bound best known solution (explores potential ~~sol~~ routes)

Algorithm :

1. Nearest Neighbor [$O(n^2)$]
2. 2-opt Heuristic
3. Lin-Kernighan Heuristic
4. metaheuristics
   - Simulated annealing

GOOD LUCK

- Genetic Algo.
- Ant colony optimization
- MST

Applications :

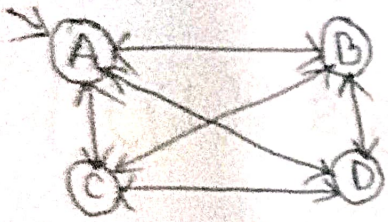
- Tsp is an NP hard problem. There's no known polynomial time solution
- Widely studied in optimization, AI, operations research.
- Delivery service
- manufactoring & production
- Network design
- DNA sequencing
- ~~vi~~ vehicle Routing
- Circit board.

GOOD LUCK

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

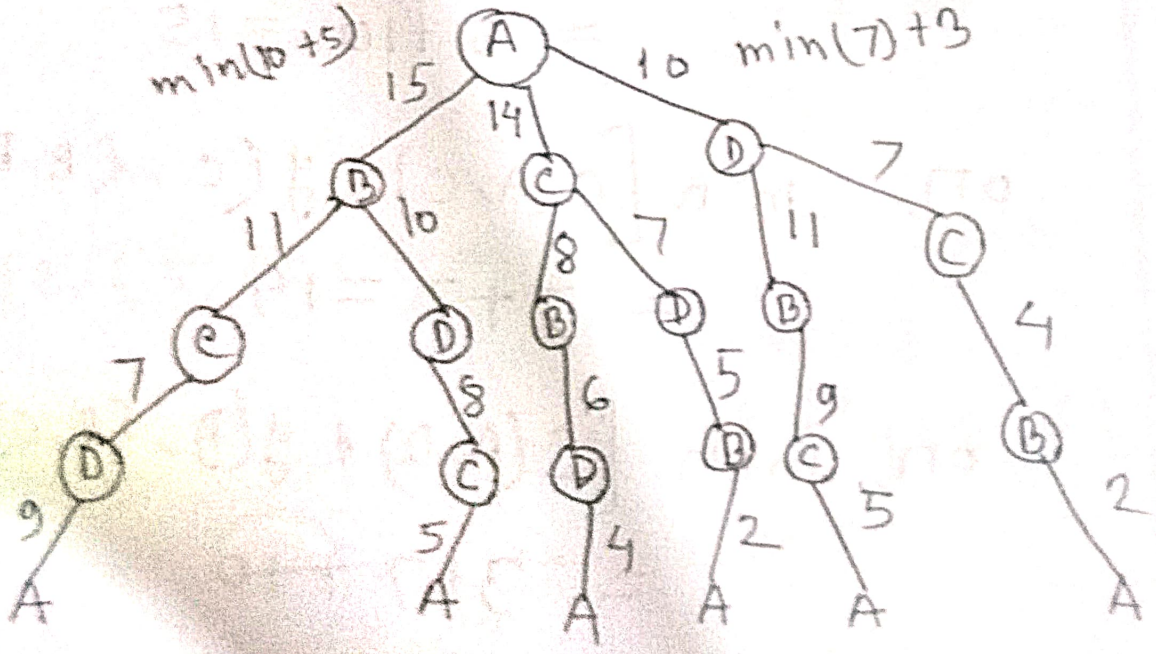
TSP (using Dynamic programming)



Cost

	A	B	C	D
A	0	5	7	3
B	2	0	4	2
C	5	2	0	3
D	9	2	3	0

Tree :



min path :

A → D → C → B → A which is 10

TOPIC NAME : DAY : TIME : DATE : / /

Algo : graph, starting, set of unvisited

$g(i, S)$

$= \min [c(i, j) + g(j, S - \{j\})]$

cost, selected one

① $\Rightarrow g(A, \{B, C, D\}) = \min [c(A, B) + g(B, \{C, D\})$

$= 5 + 10 = 15$

or, $\min [c(A, C) + g(C, \{B, D\})]$

$= 7 + 7 = 14$

or, $\min [c(A, D) + g(D, \{B, C\})]$

$= 3 + 7 + 10$

$= 10$

GOOD LUCK

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

(II) $g(B,\{C,D\}) = \min[C(B,C) + g(C,\{D\})]$

$= 4+7 = 10$

or, $\min[C(B,D) + g(D,\{C\})]$

$= 2+8 = 10$

$g(C,\{D\}) = \min[C(C,D) + g(D,\phi)]$

$= 3+4 = 7$

$g(D,\{C\}) = \min[C(D,C) + g(C,\phi)]$

$= 3+5 = 8$

(III) $g(C,\{B,D\}) = \min[C(C,B) + g(B,\{D\}]$

$= 2+6 = 8$

or, $\min[C(C,D) + g(D,\{B\})]$

$= 3+4 = 7$

$g(B,\{D\}) = \min[C(B,D) + g(D,\phi)]$

$= 2+4 = 6$

$g(D,\{B\}) = \min[C(D,B) + g(B,\phi)]$

$= 2+2 = 4$

"GOOD LUCK"

TOPIC NAME : DAY :
TIME : DATE : / /

(iv) $g(D,\{B,C\}) = \min[c(D,B) + g(B,\{C\}]$

$= 2+9 = 11$

$[c(D,C) + g(C,\{B\}]$

$= 3 + 4 = 7$

$g(B,\{C\}) = \min[c(B,C) + g(C,\phi)]$

$= 4+5 = 9$

$g(BC,\{B\}) = \min[c(C,B) + g(B,\phi)]$

$= 2+2 = 4$

So, Path,

$A \to D \to C \to B \to A$

$= 3+3+2+2 = 10$

GOOD LUCK

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

N-queens problem

Given an integer n, the task is to find the solution to the n-queens problem, where n queens are placed on an n*n chessboard such that no two queens can attack each other.

- State space : all possible n-queen configurations.
- Objective function : Num of pair wise conflicts.
- Possible level improvement stategy : Move one queen within its column to reduce conflict

Column : $C_1 = C_2$ [not safe]

Row : $R_1 = R_2$ ["]

$abs(C_1 - C_2) = abs(R_1 - R_2)$

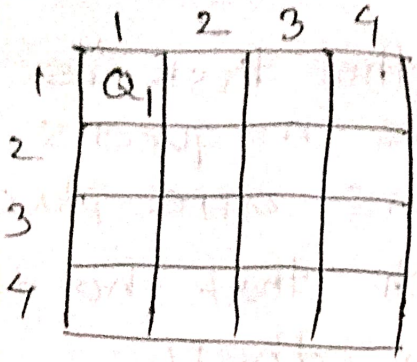
GOOD LUCK

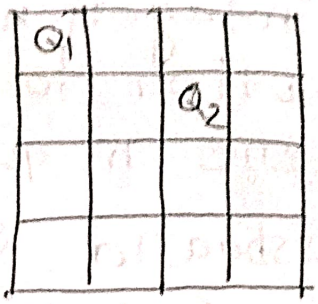
TOPIC NAME : _____ DAY : _____

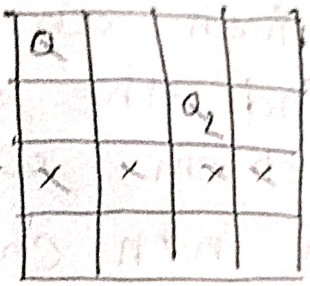
TIME : DATE : / /

4 queens

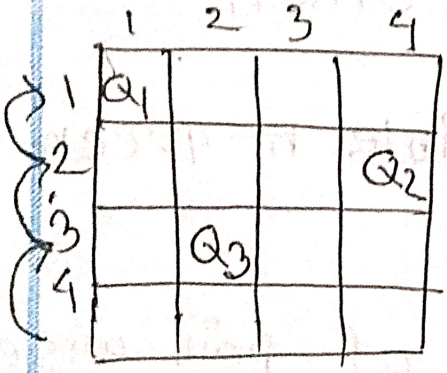
attack → column Row diagonal

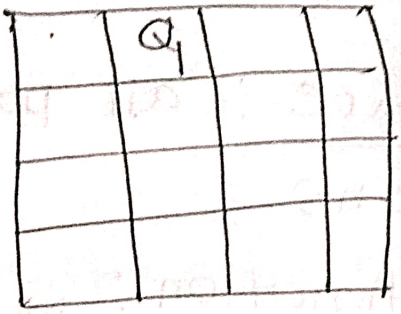


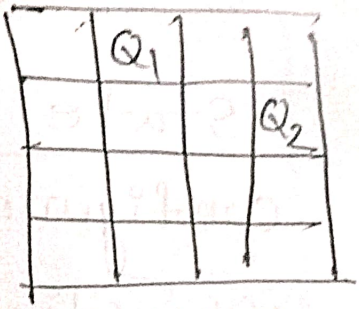


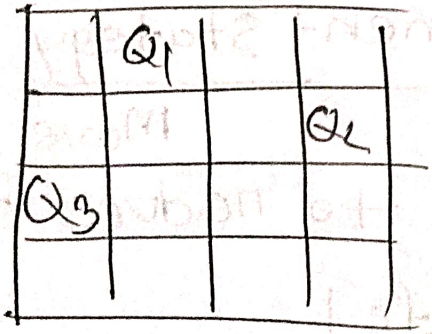


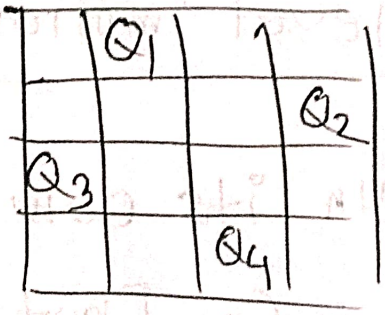
Back track











= [2, 4, 1, 3] column

GOOD LUCK™

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Hill-Climbing Search

It is a heuristic search algorithm that belongs to the family of local search methods. It's designed to solve problems where the goal is to find an optimal solution by iteratively moving from the current state to a better neighboring state according to a evaluation function.

- local search algorithm
- greedy approach
- Beam value (β) = 1
- No backtracking

Algorithm

① Evaluate the initial state in loop until a solution is found or there are no operations left.

GOOD LUCK

TOPIC NAME : _____ DAY : _____ TIME : DATE : / /

- select and apply a new operator
- Evaluate the new state
  - If goal then quit
  - If better than current state then it is new current state

* Function Hill climbing (problem) returns a state that is a local max.

Input : problem, a problem

local variables : current, a node
neighbor, a node

current ← MAKE-NODE (INITIAL-STATE [problem])

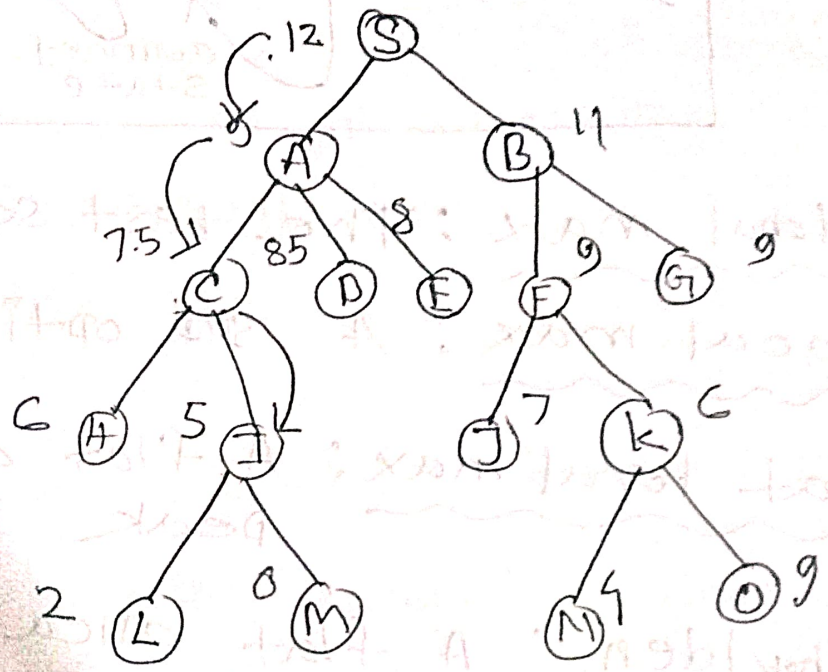
loop ← neighbor ← a highest valued successor of current

TOPIC NAME : _____ DAY : _____

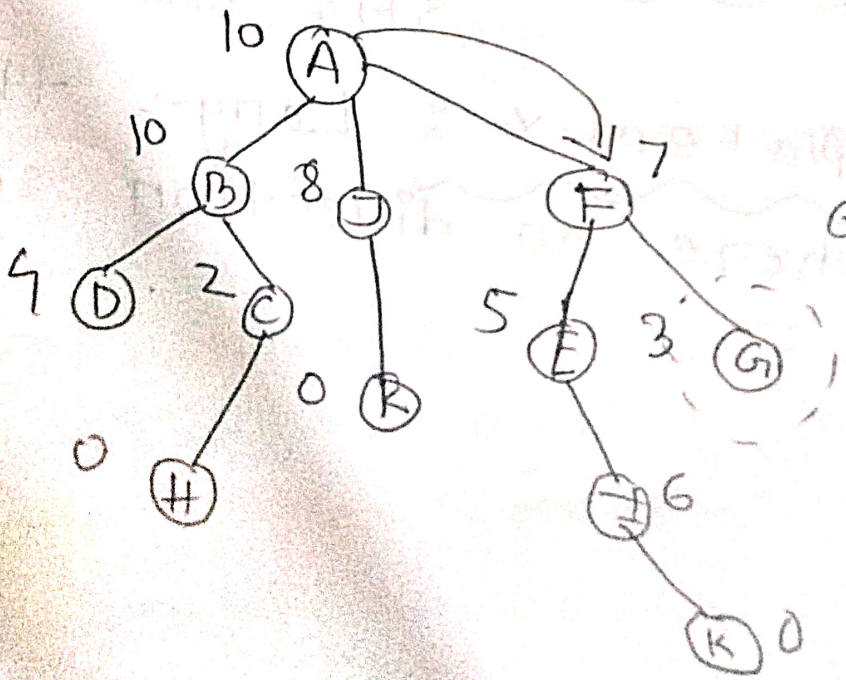
TIME : DATE : / /

if value[neighbor] ≤ value [current]
then return STATE [current]

current ← neighbor.



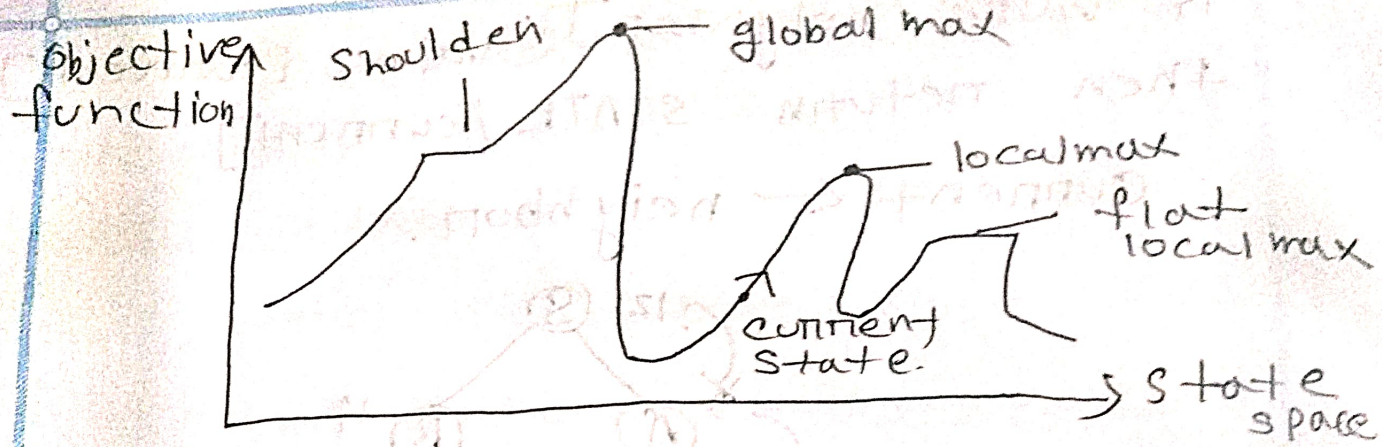
Ex:



G is local minimum

GOOD LUCK

DAY :
TOPIC NAME : TIME : DATE : / /



- Global max : The best solution
- Local max : A sub optimal peak
- Flat local max : A flat area at a peak
- Shoulder : A flat area that can still lead uphill.
- Plateaux : Large flat regions where no direction seems better.

GOOD LUCK

TOPIC NAME: _____ DAY: _____

TIME: DATE: / /

Local beam search

- Start with k random states
- At each step
  - Generate all successors of all k states
  - From this pool of successors, pick the best k overall
  - Continue until a goal is found.

k 'Greedy search in parallel

- Each search follows its own path independently
- No information sharing
- A bad starting point may waste effort.

Local beam search is not just multiple greedy searches in parallel - it's smarter because the k states compete and only the best survive each round.

GOOD LUCK