Local Search

[**]

Uninformed Search
Informed Search
} Uses backtracking because path matters there, perfect goal matttred.

[**] Uninformed & Informed searches are very expensive in large scale graphs/infinite graphs.

[**] There are problems where we are not concerned with path. Only the solution itself matters.

↳ Not concerned with perfect solution.

↳ So backtracking & path বের করার জন্য Node Sote করার, এই জিনিসগুলা বাদ দিয়ে যেই algorithm গুলা/search algorithm গুলার discuss করা হয়, তাদের local search বলে।

[**] Advantages of local search

① time কম লাগে
② Memory কম লাগে
③ Large/infinite graph ও বেশ ভালো কাজ করে।

Example: hill climbing algorithm, simulated annealing.

** Approaches of local Search:

↳ Keep a "current" state & tries to improve it.

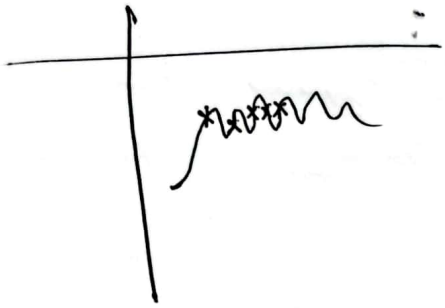
↳ Starts with an initial guess of solution and gradually improving it untill it is the optimal one.

** Disadvantages of local Search:

① Local Maxima: There can be multiple local maxima. So depending on starting point it can reach a local maxima and terminate the algorithm.
In this case it returns top point but not the best one.

② Plateaus: The space has a broad flat region. So the algorithm can not move forward or backward because the next or previous point is not leading towards anything, nor the optimal point nor the sub-optimal point. So it gets stuck.

③ **Redges:** Multiple local maximas are closed to each other. So a small step can lead to random sub optimal points.



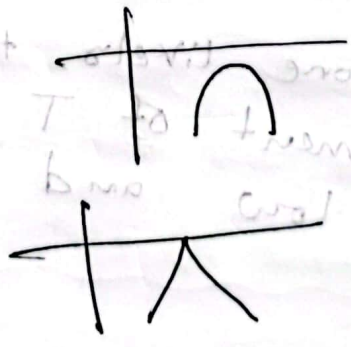
[***] **Solution:**

① **Random restart:** After reaching a top randomly restart to find the global maxima.

② **Problem reformulation:** So, we can reformulate our problem space to avoid unavoidable situations.

$$\sum (y-\hat{y})^2$$

$$\hookrightarrow |y-\hat{y}|$$



③ Simulated Annealing:

```
c = Cint    # initializing current state

for T = Tmiax to Tmin :

    Ec = E(c)
    N = Next(c)
    EN = E(N)

    ΔE = EN - Ec
    if ΔE > 0 :
        c = N
    else if (e^(ΔE/T) > rand(0,1)) :
        c = N

End
```

when T is large $e^{\Delta E/T}$ is large, so it is more likely to be True. With the decrement of T probability of 'True' becomes low and it terminates.

Informed Search

Uniform Cost Search (informed search)

- Evaluation function, $f(n) = g(n)$ → Path cost from 'start node' to 'n'

Greedy Best First Search (GBFS)

- Evaluation function, $f(n) = h(n)$ → heuristic

GBFS Optimal ?

- Not optimal. Because এটি True path cost consider করে না, approximate একটি heuristic consider করে, So approximate একটি sob path দেয়।

GBFS Complete ?

- Tree: Not complete. visited / expanded track রাখা হয় না, so একটি Node বারবার push হয়, loop এ পড়ে যায়।
  In case of both finite and infinite tree it is not complete.
- Graph: Complete for finite graph. loop prevent করা যায় but for infinite it's not complete.

Time and Space Complexity ?

- $O(b^m)$ (same as bfs, dfs)

Practical implementation এ heuristic এর quality এর উপর Time & Space Complexity নির্ভর করে।

A* Search

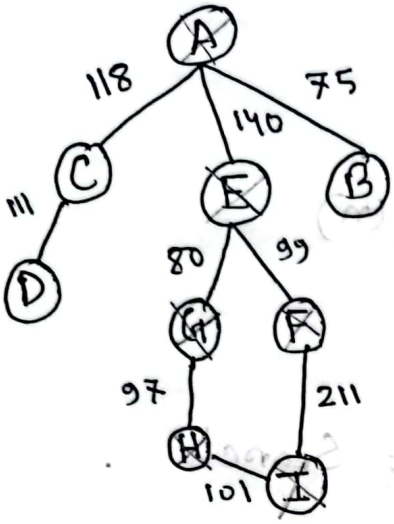
- Evaluation function, $f(n) = h(n) + g(n)$
- when $h(n) = 0$:

  A* Search = Uniformed Cost Search .

  when $g(n) = 0$:

  A* Search = GBFS

A* Search Simulation



State	h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

Soln:

A^{0+366}

A^{366} → $C^{118+329=447}$, $E^{140+253=393}$, $B^{75+374=449}$

E^{393} → C^{447}, B^{449}, ~~G~~ $^{220+193=413}$, $F^{239+178=417}$

G^{413} → C^{447}, B^{449}, F^{417}, ~~H~~ $^{317+98=415}$

H^{415} → C^{447}, B^{449}, ~~F^{417}~~, $I^{418+0=418}$

F^{417} → C^{447}, B^{449}, ~~I^{418}~~, ~~$I^{450+0=450}$~~

I^{118}

Parent	Node	Distance from start, g(n)
–	A^{366}	0
A^{366}	C^{447}	0+118=118
A^{366}	E^{393}	0+140=140
A^{366}	B^{449}	0+75=75
E^{393}	G^{413}	140+80=220
E^{393}	F^{417}	140+99=239
G^{413}	H^{415}	220+97=317
H^{415}	I^{418}	317+101=41
F^{417}	I^{450}	239+211=450
I^{118}		

Path: $A^{366} \rightarrow E^{393} \rightarrow G^{413} \rightarrow H^{415} \rightarrow I^{118}$

Cost:

(**) Admissibility:

- $h(n) \leq C(n, \text{Goal})$

→ A graph's heuristic is admissible when this condition is true for all the nodes.

If one Node breaks this condition, the graph's heuristic is not admissible. and whe the condition breaks we say "h" is overestimated".

- When heuristic is not admissible, it gives sub-optimal path.

Example:

E

→ $h(E) = 253$

→ $C(E, \text{Goal}) = \text{lowest}(278, 310) = 278$

$253 \leq 278$ ✓

A* Search optimal ?

- efficient path that for or

- if heuristic is Admissible:

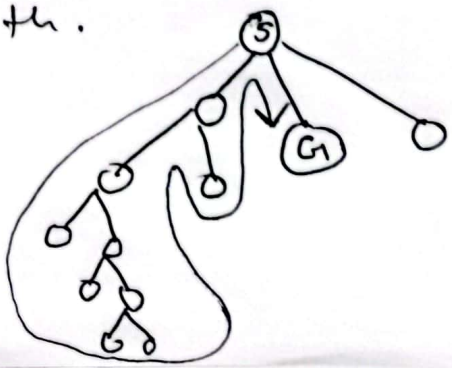
  Optimal

  else:

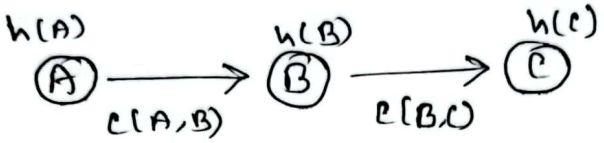
  Not Optimal

when h(n) is "optimal" but h(n) is not "consistante" then it takes more time to find the optimal path.

when h(n) is "Optimal" but "not consistante". It surely gives optimal path but take more time. Roams around unwanted paths. then finds optimal path.



Consistancy:

- $h(n) \leq c(n, n') + h(n')$
- 

  ↳ $h(A) \leq c(A,B) + h(B)$
  ↳ $h(B) \leq c(B,C) + h(C)$

→ $h(n)$ is "consistante" is the condition is true for all the nodes. (except Goal)

→ $h(n)$ can be Admissible but not consistante

↳ তখন time & Space বেশি লাগে।

A* Search Complete? (← loop এ পড়ে কিনা, unwanted nodes এ ঘুরাঘুরি করে কি না)

- Graph Search: if $h(n)$ Admissible:
  Complete
  else:
  Not Complete

- Tree Search: if $h(n)$ is Admissible & Consistante:
  Complete
  else:
  Not Complete