AI
Posted By: Faisal Habib
AI, as studied this days may not be homogeneous. Technics were developed and they do work. But there are too many discrepancies in present theories-technics-definitions-applications-such. Single salesman problem is considered to be the basic of AI technics. Think multiple salesmen released from different nodes. Parent nodes can have same weight, they may vary. But considering a domain with some 'parent' nodes of same weight -- if one agent is released at every parent node, certainly calculation efficiency will increase. Can we calculate RMS of such a problem?!?
This much for now.

Latest Reply:
What if two salesmen get released from every parent node? One will reach for max weight and the other for the min weight. And then they will search for second max and second min.
so, we choose parent node(s) like this::
>> after completing a breadth first search to get a basic idea of the problem domain (considering the problem domain is updatable)
1. How many parent node we need?
2. What r their weights?
3. What is the max weight nearest to each parent node chosen?
4. What is the min weight nearest to each parent node chosen?
5. When we have max and min weights identified, do we expand -- how to expand? min or max to choose from??
faisalH
Reply From: Faisal Habib Aug 03, 2016 1:47PM

What can be a breadth first search for multiple salesmen problem::
1. we have a map
2. we know our strong points >> nodes we think important
3. we need to reach selected nodes from different parent nodes >> unlike single SM prob.
4. So, considerations start like this::
a. study the map and [calculate no of nodes] →

I. Identify no of node(s) of a specific weight
II. identify how many parent nodes we need to consider [let us consider one → the six node]
III. we release six SM
IV. go to every reachable and freeze [calculation: if white then add, if yellow then multiply]
V. choose top weight
VI. select TOP as PN
VII. go to IV and iterate. [i=6]
VIII. When i=6 choose next TOP as PN [NOTE: searching way to my pavilion]

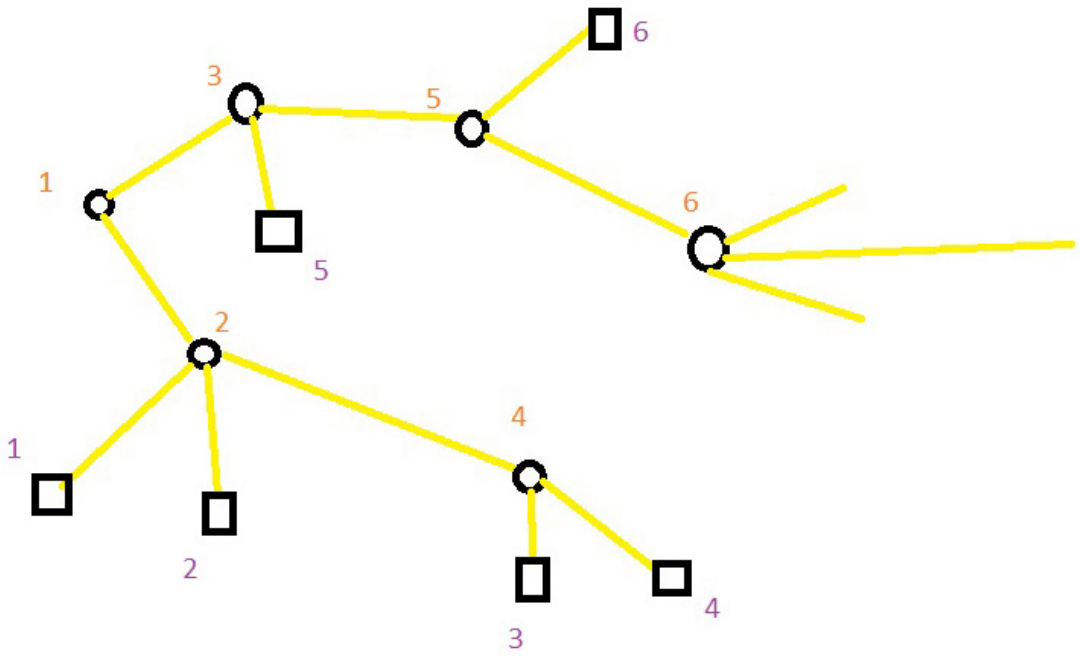
//b. we put same weight for each parent node chosen }}last updated 07-08-16{{
c. we calculate distances from each parent node to the destination of the agent considering that one SM get released from each PN.
d. we get weights of destination nodes
e. now destination nodes become PN
f. repeat until mapping is complete.
g. first run over
h. start readjustment of weights >> ???
faisalH

Reply From: Faisal Habib Aug 03, 2016 2:02PM

Update:: 12-08-16

Tree search.

A tree generated. First 6 nodes, 6 leaves and three open.



Algorithm generated – following::

Start::

Assumption: every node has same weight and every connection has same weight.

Select N1 → PN1

2SM → N2 & L3 [calc → let 2 is top]

Select N2 as PN2

3SM → N$ & L1, L2

Select N4 as PN4

2SM to L3, L4 freeze

Back to Pav. [recursion]

Select N3 as PN3

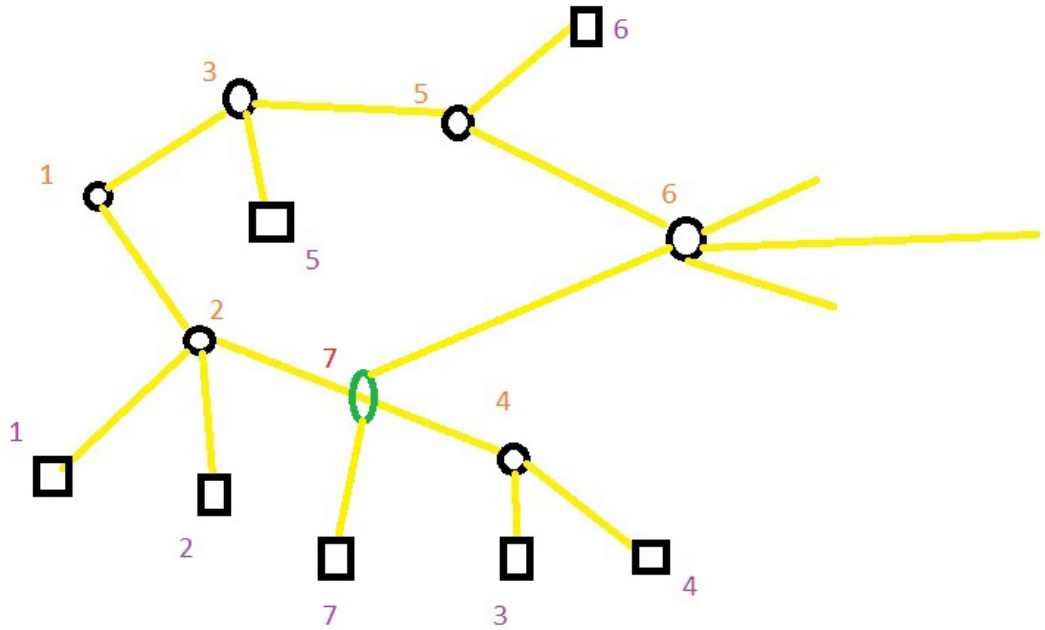
2SM → N5 & L5

Select N5 as PN5

2SM → N6 & L6

Select as PN6 --- it is Pav2.

Second tree generated as ::



N7 is an interactionNode

The algorithm – following::

N1 → PN1

2SM → N2 & N3 [calc → let N2 is TOP]

Select N2 → PN2

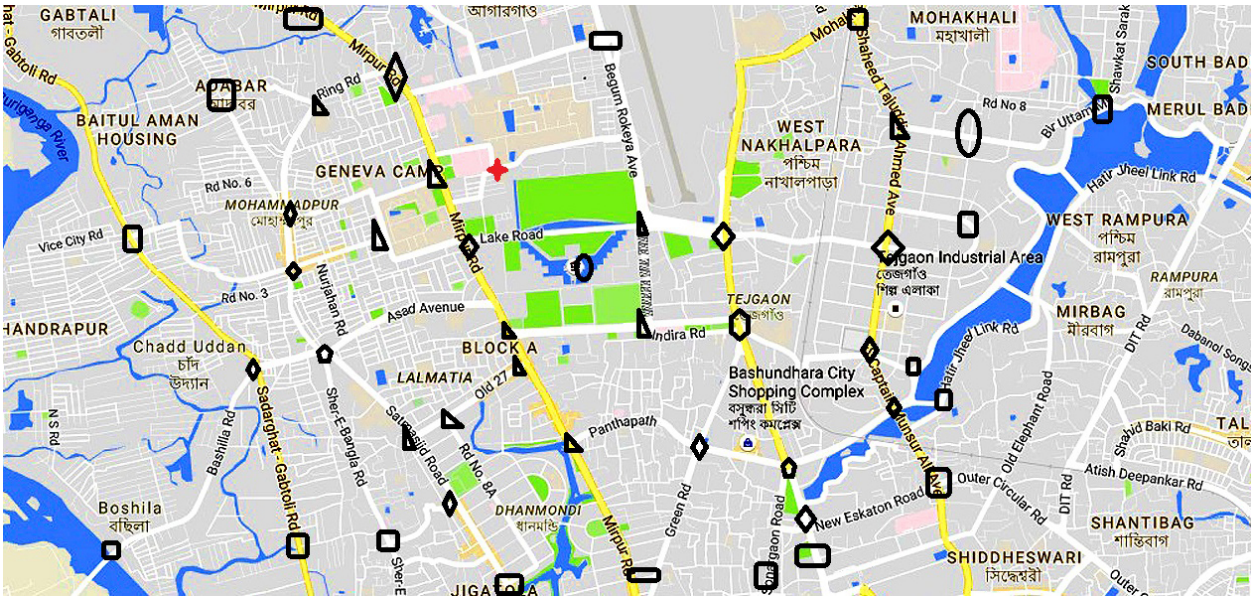
3SM → N7 & L1, L2

Select → N4, N6, L6 [calc → let N6 as PN4]

Select N6 as PN4

4SM → N5 & --- ???

Now, if we revise the dAIs with treeSearchAlg with same weight distributions, it is possible to translate it into a tree search, which is more efficient.



Now we need to solve the dAIs with interactionNode algorithm, which is an updatable weight algorithm. So, a general tree search is actually a uniform-cost search of a graph. The interactionNode algorithm is a variableCost graph search. The variableCost graph search can be simulated with iterativeDeepening search (IDS) algorithm. Let us choose iterativeRecursionAlgorithm (IRA) as a replacement of iterativeDeepening Algorithm – which following::

1. Go to N6 freeze
2. To Pav
3. Go to N4, L7
4. To Pav [calc → let N6 is Top]
5. Go to line no 1 → replace and continue.

Next analysis is variable iterative recursion Algorithm (VIRA) – which is an updatable weight algorithm (UWA).

A probabilistic search:

Let we r at 6 node. We need cooler temp. to reach 3 node time required is 0.75hr, to reach 4 node time req is 1 hr. to reach 5 node time req is 1.25 hr. so, if we have data with resolution/min for a day-month-year we can predict wind-temp at any node better with res/day, than res/month and res/yr. probability as calculated from Sparso data provided it is at 6 node and we have sensors at Sparso & every reachable node.

Genetic algorithm skipped.

Nondeterministic environment search [and-or graph search]

Suppose we don't have any data in Sparso database, but we have sensors installed at every node. Then we read sensors for present wind-temp condition(s) and decide if WT5 <= WT4 <= WT3 then head to 5node. If not decide again and head to.

Online search agent (OSA):

Online means we have working sensors at every node. We take our agent as a database updater. The agent takes input from sensors of wind speed and temp and updates database(s) at per second resolution. Let tempN5S>tempN6>tempN4N and wind speed is such that Compared with previous data, we find that probabilistic search (PS) gives us N5 is TOP.

Target [at this moment]:

We r heading home, LN Jigatola, selecting coolest temps. And, we decide to travel downhill.

I.RTA* skipped.