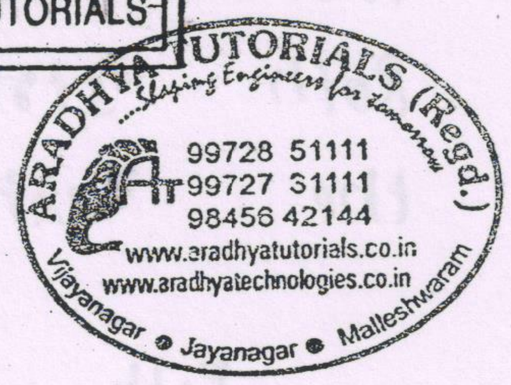
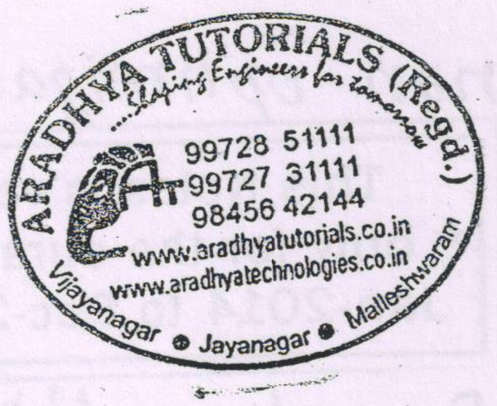
UNIT - VII

COPING WITH LIMITATIONS OF ALGORITHMIC POWER -



- n Queens Problem
- Hamiltonial circuit Problem
- Subset - Sum problem.
- Brand and Bound
  - Job Assignment Problem
  - Knapsack Problem
  - Travelling salesperson Problem

- Approximation Algorithms
  - Travelling salesperson Problem
  - Knapsack problem



What is Branch and Bound? How is it different from backtracking? Explain

Both Backtracking and Branch and Bound are a algorithm design techniques that make it possible to solve atleast some large instances of difficult combinatorial problems. Both strategies are an improvement over the exhaustive search technique.

Both backtracking and Branch and Bound are based on the construction of the state space tree, whose nodes reflect specific choices made for a solution's components. Both the techniques terminate a node as soon as it is realized that it does not produce a solution.

Inspite of the similarities between the 2 techniques, yet there are certain differences such as -

(i) Branch and Bound is applicable to optimization problems whereas Backtracking is applicable to non-optimization problems.

(ii) For Branch and Bound, the state space tree is normally developed in the BFS order whereas for

backtracking it is normally generated in the DFS order.

(iii) Problems such as Job-assignment problem, travelling salesman problem, knapsack problem can be solved using Branch and Bound technique whereas n-queens problem, hamiltonian circuit problem, sum-of-subset problems can be solved in backtracking technique.

What is Backtracking? Apply the same to n-queens problem and explain.

Backtracking is an algorithm design technique which is an improvement over the exhaustive search technique. It is based on the construction of a statespace tree. The nodes of a tree reflects the specific choice made by the programmer while finding the solution. If there is no legitimate option for the next candidate, then the progress along that node is terminated. In such a case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option.

The root of the state-space tree represents an initial state before the search for a solution begins. The nodes of the first level in the tree represents the choices made for the first component of a solution, the nodes at the second level represents the choices made for the second component and so on. A node in a state-space tree is said to be "promising" if the partially constructed solution may still lead to the final solution. Otherwise it is "non promising".

The N-Queens problem refers to the problem of placing n-queens on a n by n chess board such that no 2 queens attack each other by being in the same row or in the same column or on the same diagonal.

Consider a 4 queen problem. We start with an empty board and then place queen 1 in col 1 of first row. We then place the second queen after trying unsuccessfully in col 1 and 2 in the square (2,3). This proves to be a dead end since there is no acceptable position for queen 3. So, the algorithm backtracks and places queen 2 in the next acceptable

position i.e square (2,4). Then queen 3 is placed at square (3,2) which proves to be another dead end. The algorithm then backtracks all the way to queen 1 and moves it to square (1,2). Queen 2 then goes to (2,4), queen 3 to (3,1) and queen4 to (4,3). This can be represented in the form of a state-space-tree as shown below -

(Refer class notes)

What is Branch and Bound Method? Explain the concept with an example -

Branch and Bound is an algorithm design technique which is an improvement over the exhaustive search technique. It is based on the construction of a state-space-tree. The nodes of a tree reflects the specific choice made by the programmer while finding the solution. If there is no legitimate option for the next candidate, then the progress along the node is terminated.

P.T.O.

Branch and Bound requires 2 additional items -

(i) A bound value associated with every node of the state-space-tree

(ii) the value of the best solution seen so far.

Consider the following example -

(give the job assignment problem ex given in class notes)

Hamiltonial Circuit Problem -

(No Theory. For problems refer class notes)

Subset - sum problem -

The subset-sum problem can be solved by using a state-space tree. The state-space tree can be constructed as a binary tree. The root of the tree represents the starting point. Its left and right children represent respectievely the inclusion and exclusion of s1. Similarly going to the left from a node of the first level corresponds to the inclusion of s2 while going to the right corresponds to its exclusion.

Thus a path from the root to a node on the i^{th} level of the tree indicates which of the first i nos have been included in the subsets represented by that node.

Eg - (Refer class notes)

Give the approximation algorithm for the travelling salesman Problem -

The approximation algorithm for the travelling salesman problem include -

1. Nearest - Neighbour Algorithm
2. Multi Fragment - heuristic Algorithm
3. Minimum Spanning Tree Based Algorithm

1. Nearest Neighbour Algorithm

Step 1 Choose an arbitrary city as the start.

Step 2 Repeat the following operation until all the cities have been visited: go to the unvisited city nearest the one visited last (ties can be broken arbitrarily).

Step 3 Return to the starting city.

2. Multi-fragment heuristic Algorithm

Step 1 Sort the edges in increasing order of their weights. (Ties can be broken arbitrarily.) Initialize the set of tour edges to be constructed to the empty set.

Step 2 Repeat this step until a tour of length n is obtained, where n is the number of cities in the instance being solved: add the next edge on the sorted edge list to the set of tour edges, provided this addition does not create a vertex of degree 3 or a cycle of length less than n; otherwise, skip the edge.

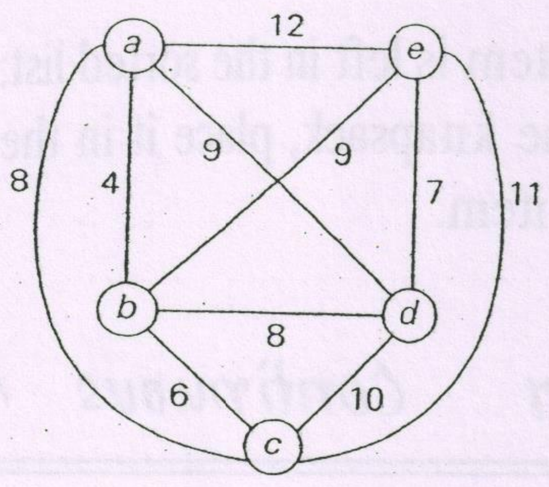
Step 3 Return the set of tour edges.

3. Minimum spanning Tree Based algorithm-

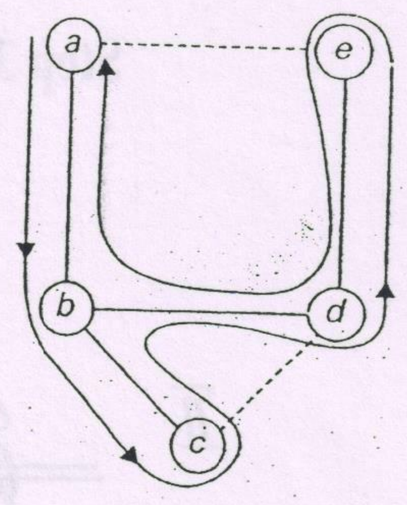
Step 1 Construct a minimum spanning tree of the graph corresponding to a given instance of the traveling salesman problem.

Step 2 Starting at an arbitrary vertex, perform a walk around the minimum spanning tree recording all the vertices passed by. (This can be done by a DFS traversal.)

Step 3 Scan the vertex list obtained in Step 2 and eliminate from it all repeated occurrences of the same vertex except the starting one at the end of the list. (This step is equivalent to making shortcuts in the walk.) The vertices remaining on the list will form a Hamiltonian circuit, which is the output of the algorithm.



(a)



(b)

ADA 7

give the Approximation Algorithm for Knapsack problem -

The approximation Algorithm for Knapsack problem are -

I greedy Algorithm for discreet Knapsack

Step 1 Compute the value-to-weight ratios $r_i = v_i/w_i$, $i = 1,\ldots,n$, for the items given.

Step 2 Sort the items in nonincreasing order of the ratios computed in Step 1. (Ties can be broken arbitrarily.)

Step 3 Repeat the following operation until no item is left in the sorted list: if the current item on the list fits into the knapsack, place it in the knapsack; otherwise, proceed to the next item.

II greedy Algorithm for Continuous Knapsack

Step 1 Compute the value-to-weight ratios v_i/w_i, $i = 1,\ldots,n$, for the items given.

Step 2 Sort the items in nonincreasing order of the ratios computed in Step 1. (Ties can be broken arbitrarily.)

Step 3 Repeat the following operation until the knapsack is filled to its full capacity or no item is left in the sorted list: if the current item on the list fits into the knapsack in its entirety, take it and proceed to the next item; otherwise, take its largest fraction to fill the knapsack to its full capacity and stop.

This Notes is valid
★ only for the duration ★
Jan-2014 to Dec-2014