SNS COLLEGE OF TECHNOLOGY,
COIMBATORE –35

Approximation Algorithms for NP Hard Problems

Approximation Approach

- Difficult optimization problems (i.e., there are no known polynomial-time algorithms for these problems) can be solved approximately by a fast algorithm (i.e., polynomial-time algorithm). The found solution is approximate one, but it is close to the exact solution s^*.
- We would like to know how accurate the approximate solution s_a is. The accuracy of an approximate solution s_a is measured by ***accuracy ratio*** $r(s_a)$.

Approximation Approach

- For minimization problem (i.e., minimize an objective function f):

$$r(s_a) = \frac{f(s_a)}{f(s^*)},$$

where s^* is an exact solution to the problem.

Approximation Approach

- For maximization problem (i.e., maximize an objective function f):

$$r(s_a) = \frac{f(s^*)}{f(s_a)}$$

where s^* is an exact solution to the problem.

- ***Performance ratio*** of the algorithm A, denoted R_A, is the lowest upper bound of $r(s_a)$ on all instances.

Nearest Neighbor Algorithm for TSP

Step 1 Choose an arbitrary city as the start.

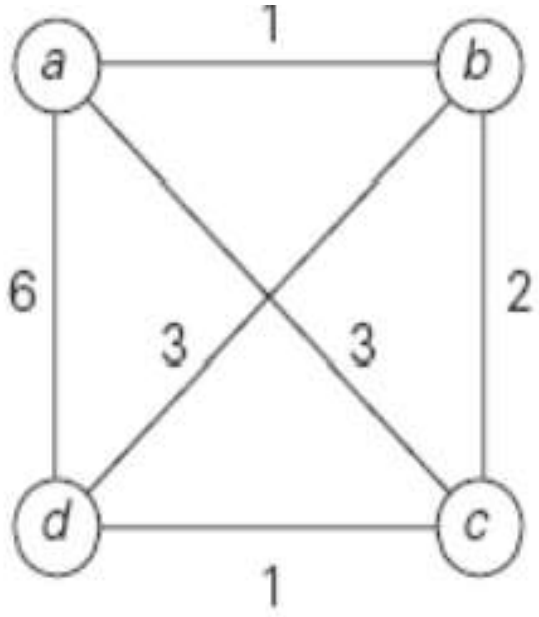
Step 2 Repeat the following operation until all the cities have been visited: go to the unvisited city nearest the one visited last.

Step 3 Return to the starting city.

Nearest Neighbor Algorithm for TSP

s_a: a - b - c - d - a of length 10

s^*: a - b - d - c - a of length 8



$$r(s_a) = \frac{f(s_a)}{f(s^*)} = \frac{10}{8} = 1.25$$

(i.e., tour s_a is 25% longer than the optimal tour s^*)

Multifragment-Heuristic Algorithm

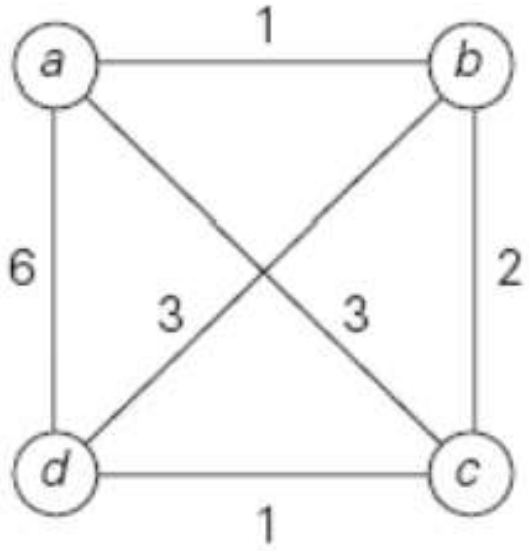
Step 1: Sort edges in increasing order of weights.
Initialize the set of tour edges to be constructed to empty set

Step 2: Add next edge on the sorted list to the tour, skipping those whose addition would've created a vertex of degree 3 or a cycle of length less than n. Repeat this step until a tour of length n is obtained.

Step 3: Return the set of tour edges

Multifragment-Heuristic Algorithm

s_a: $\{(a, b), (c, d), (b, c), (a, d)\}$



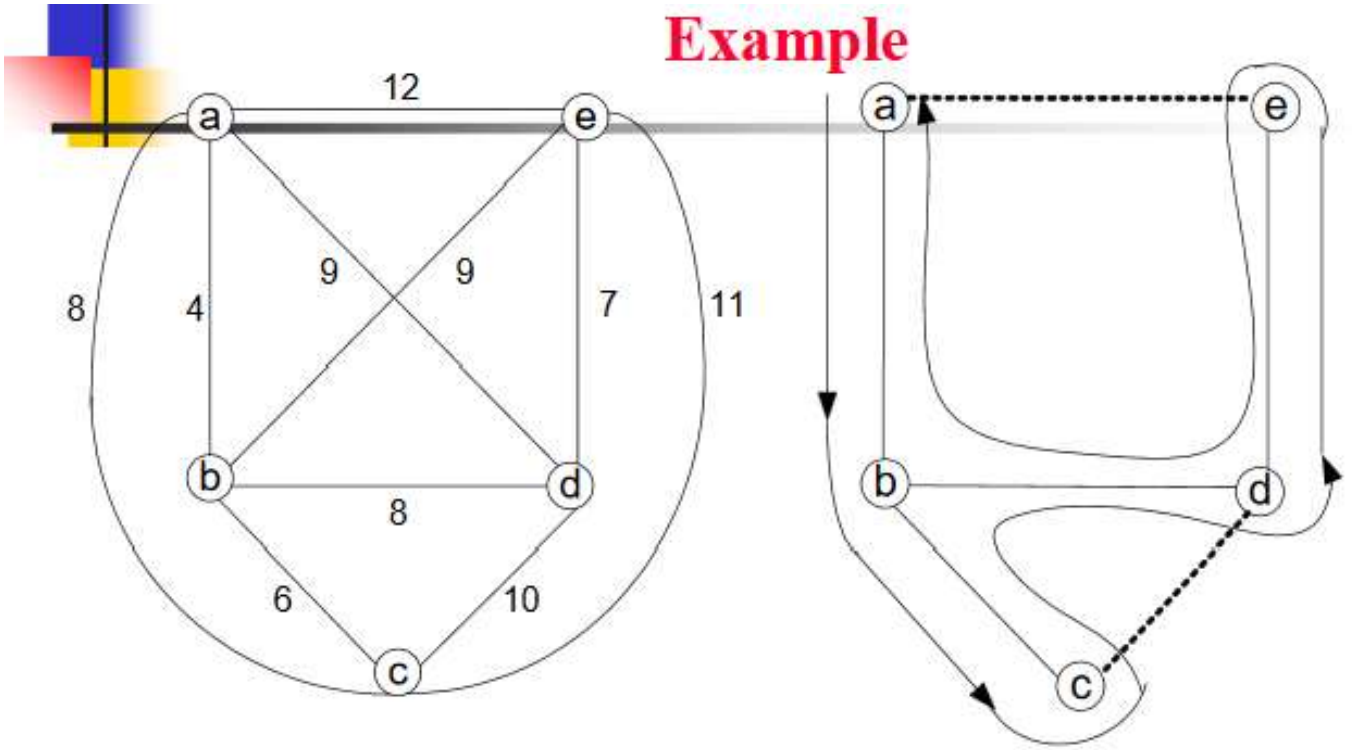
Twice Around the Tree Algorithm

Step 1: Construct a minimum spanning tree of the graph (e.g., by Prim's or Kruskal's algorithm)

Step 2: Starting at an arbitrary vertex, perform a walk around the minimum spanning tree recording all the vertices passed by.

Step 3: Eliminate from the recorded list all repeated occurrences of the same vertex except the starting one at the end of the list. The vertices remaining on the list will form a Hamiltonian circuit.

Twice Around the Tree Algorithm



Walk: a – b – c – b – d – e – d – b – a

Tour: a – b – c – d – e – a of length 39

- The tour obtained is not optimal

(better solution: 4 + 6 + 11 + 7 + 9 = 37).