TSP HEURISTICS

- Although ingenious exact algorithms for the TSP (e.g., branch and cut) have been proposed, most encounter problems with storage and/or running time for cases with more than 500 nodes

- The largest TSP instance known to have been solved to optimality involved approximately 25,000 nodes and was solved on a cluster of 96 dual processor Intel Xeon 2.8 GHz workstations. The cumulative CPU time was approximately 90 CPU years on a single Intel Xeon 2.8 GHz processor.

WHY STUDY TSP HEURISTICS

- Exact procedures are exponential.
- Heuristics are much faster (polynomial time) and more practical.
- Many such heuristics can be applied to other problems.

ASSUMPTIONS

- $c_{ij} = c_{ji}$ (symmetry)
- $c_{ij} + c_{jk} \geq c_{ik}$ (triangle inequality)
- Filled in cost matrix (the network is fully connected)

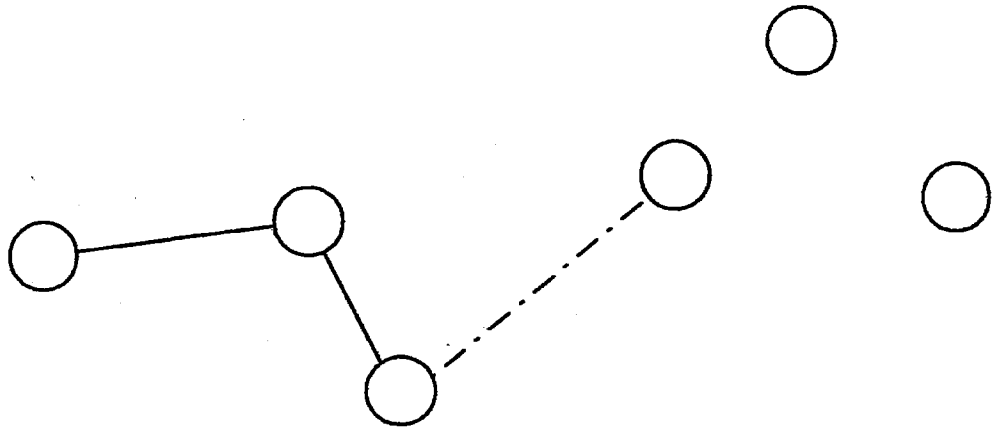
TOUR CONSTRUCTION HEURISTICS

NEAREST NEIGHBOR PROCEDURE

Step 1. Start with any node as the beginning of a path.

Step 2. Find the node closest to the last node added to the path.

Step 3. Repeat Step 2 until all nodes are contained in the path. Then, join the first and last nodes.



NEAREST INSERTION PROCEDURE

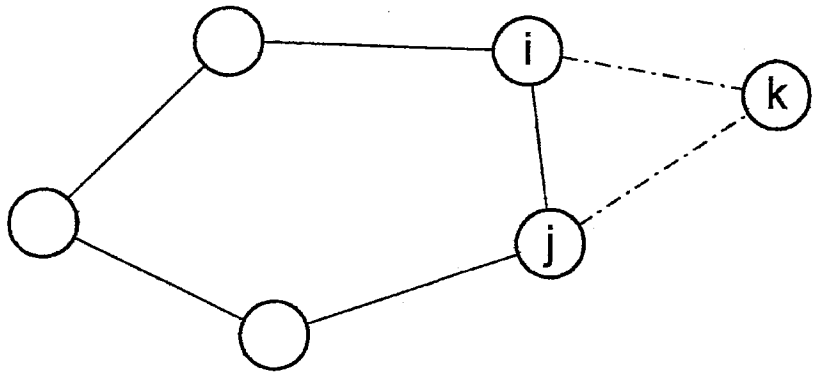
Step 1. Start with a subgraph consisting of node i only.

Step 2. Find node k such that c_{ik} is minimal and form the subtour i - k- i .

Step 3. Selection step. Given a subtour, find node k not in the subtour closest to any node in the subtour.

Step 4. Insertion step. Find the arc (i, j) in the subtour which minimizes $c_{ik} + c_{kj} - c_{ij}$. Insert k between i and j.

Step 5. Go to Step 3 unless we have a Hamiltonian cycle.



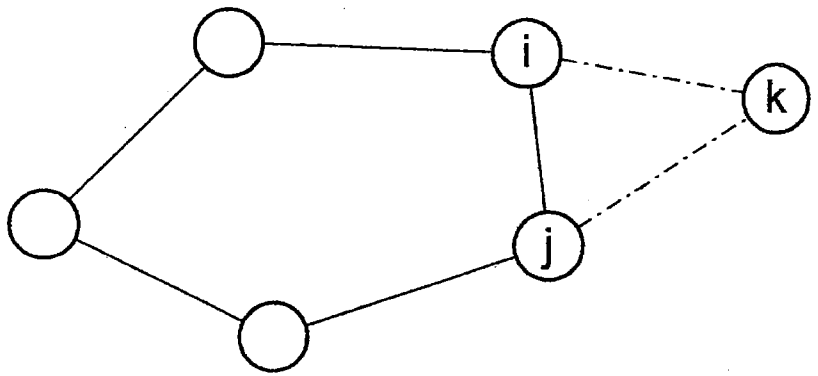
CHEAPEST INSERTION PROCEDURE

Step 1. Start with a subgraph consisting of node i only.

Step 2. Find node k such that c_{ik} is minimal and form the subtour i - k- i .

Step 3. Find (i, j) in subtour and k not, such that $c_{ik} + c_{kj} - c_{ij}$ is minimal and, then, insert k between i and j.

Step 4. Go to Step 3 unless we have a Hamiltonian cycle.



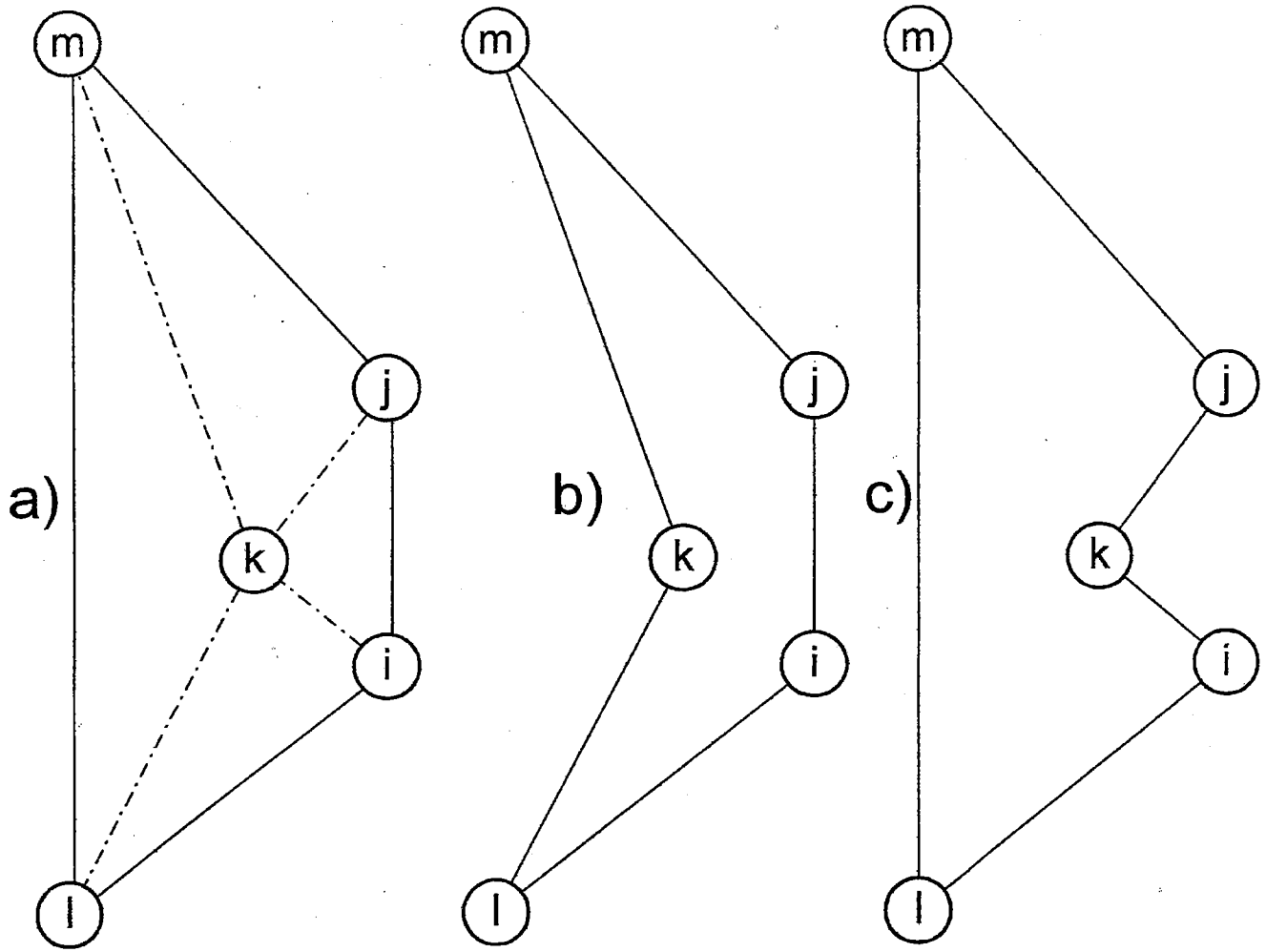
GREATEST ANGLE INSERTION PROCEDURE

Step 1. Form the convex hull of the set of nodes. The hull gives an initial subtour.

Step 2. Choose the node k* not in the subtour and the arc (i*, j*) in the subtour such that the angle formed by the two arcs (i*, k*) and (k*, j*) is largest possible.

Step 3. Insert node k* in subtour between nodes i* and j*.

Step 4. Repeat Steps 2 and 3 until a Hamiltonian cycle is obtained.



Comparison of insertion criteria: **(a)** Initial subtour with potential insertions. **(b)** Greatest angle insertion. **(c)** Cheapest insertion.

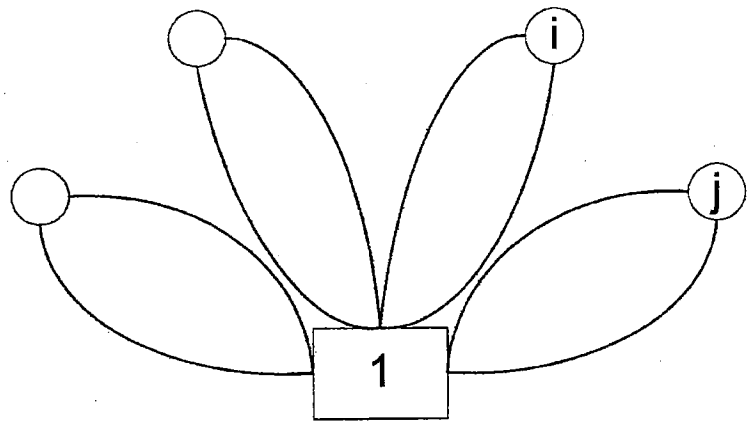
CLARKE AND WRIGHT SAVINGS PROCEDURE

Step 1. Select any node (say 1) as the starting node.

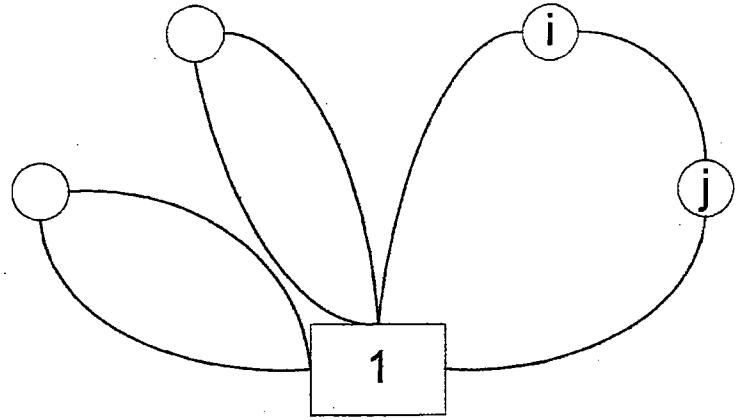
Step 2. Compute the savings $s_{ij} = c_{1i} + c_{1j} - c_{ij}$ for $i, j \in N - \{1\}$.

Step 3. Order the savings from largest to smallest.

Step 4. Starting at the top of the savings list and moving downward, form larger subtours by linking subtour endpoints i and j. Repeat until a tour is formed.



Initial configuration



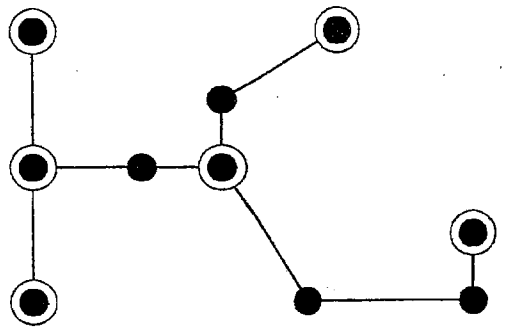
Nodes i and j are linked

CHRISTOFIDES' HEURISTIC

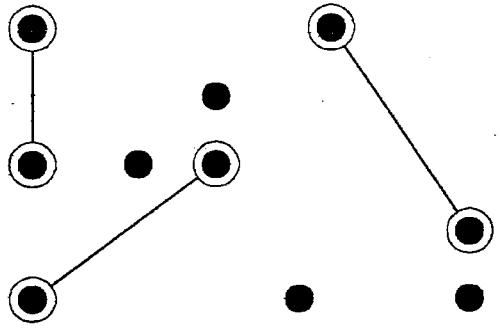
Step 1. Find a minimum spanning tree T of G.

Step 2. Identify all the odd-degree nodes in T. Find a minimum cost perfect matching on the odd-degree nodes using the original cost matrix C. Add the arcs from the matching solution to the arcs already in T, obtaining an Eulerian cycle.

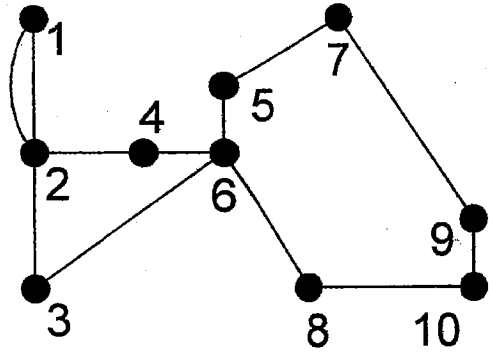
Step 3. Remove all arcs but two incident to nodes with degree greater than 2 by exploiting the triangle inequality, thereby transforming the Eulerian cycle into a Hamiltonian cycle.



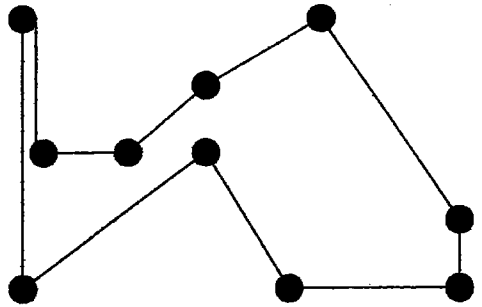
(a)



(b)



(c)



(d)

a. MST with odd nodes encircled.

b. Shortest matching of odd nodes.

c. MST plus matching = Eulerian graph.

d. Resulting tour.

PERFORMANCE OF TOUR CONSTRUCTION PROCEDURES

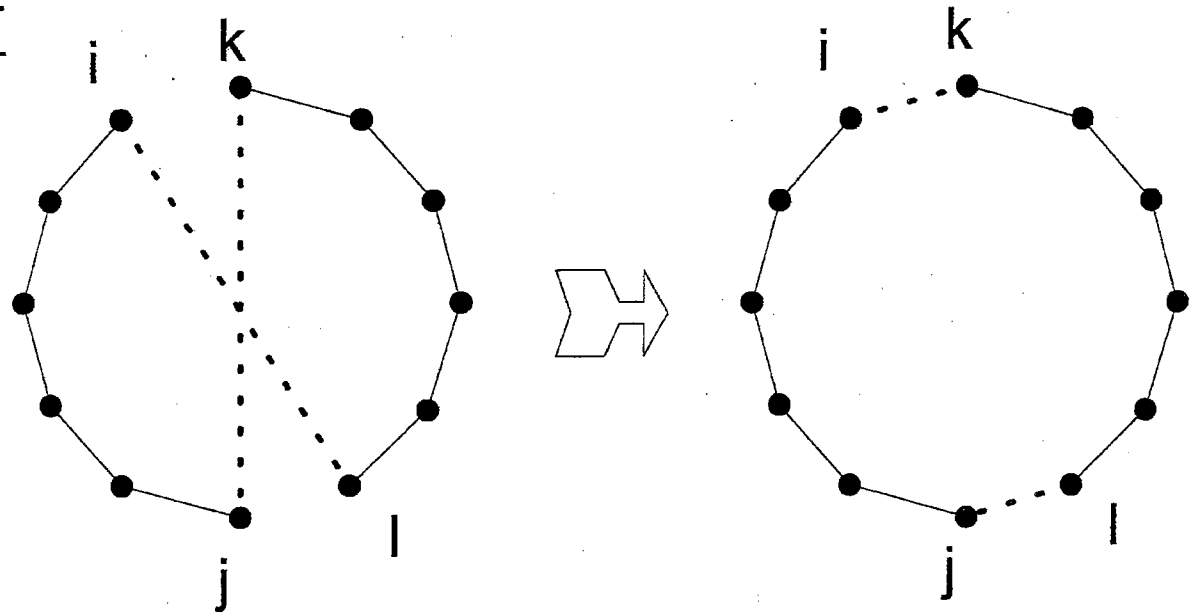
- A recent article compares popular tour construction procedures on a set of 30 Euclidean TSPs from the literature with optimal solutions. They range in size from 105 to 2392 nodes. Surprisingly, the savings heuristic is the best of those tested.

Quality of tour construction procedures

Heuristic	Average percent above optimality
Nearest neighbor	24.2
Nearest insertion	20.0
Christofides'	19.5
Modified nearest neighbor	18.6
Cheapest insertion	16.8
Random insertion	11.1
Farthest insertion	9.9
Savings	9.8
Modified savings	9.6

TOUR IMPROVEMENT HEURISTICS

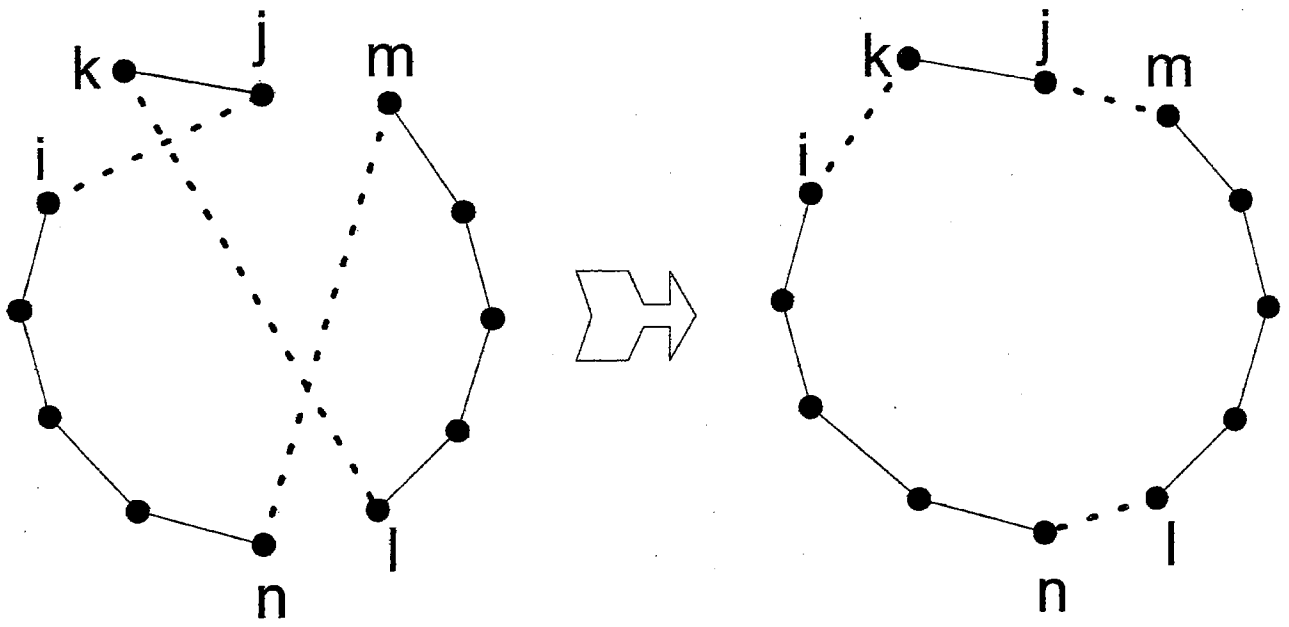
TWO-OPT



IMPLEMENTATION ISSUES

- First improvement
- Best Improvement
- Compromise improvement
- Accept uphill moves

THREE-OPT



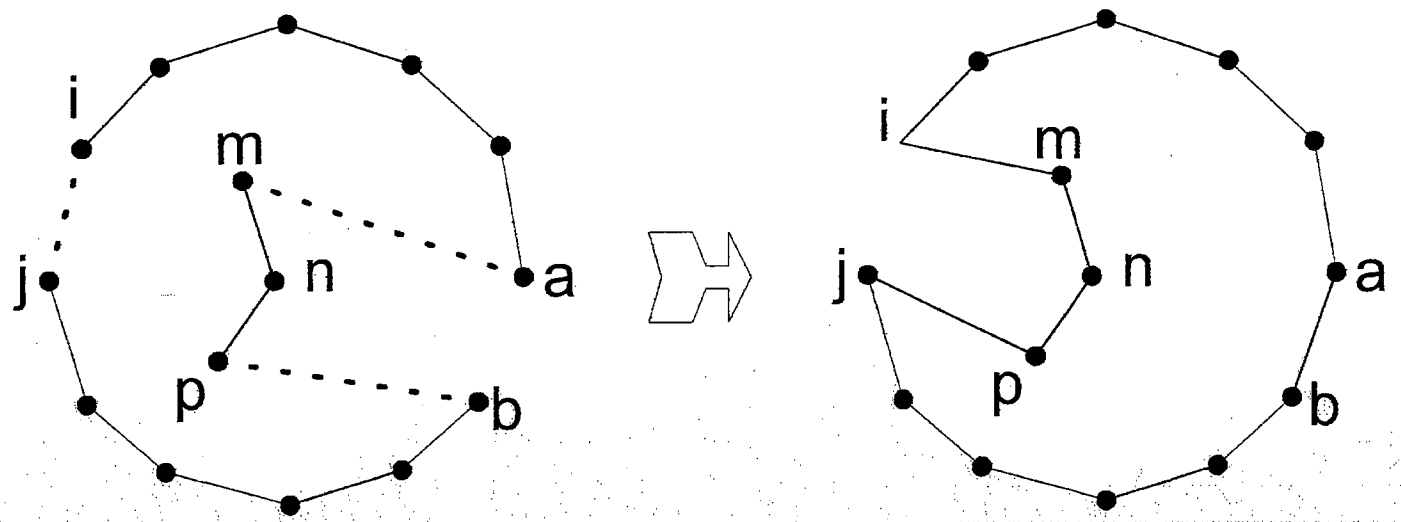
THE OROPT EXCHANGE PROCEDURE

- For each connected string of s nodes in the current tour (s equals 3 first, then 2, then 1), test to see if the string can be relocated between two other nodes at a reduced cost. If it can, make the appropriate changes.

- Consider all strings of three nodes, all strings of two nodes and then all strings of one node. When no further exchanges improve the solution, the algorithm terminates.

■ In practice, three-opt requires about twenty times as much computer execution time as does OROPT on a 100-node problem. Solutions are comparable in terms of accuracy.

OROPT



PERFORMANCE OF SOME TOUR IMPROVEMENT PROCEDURES

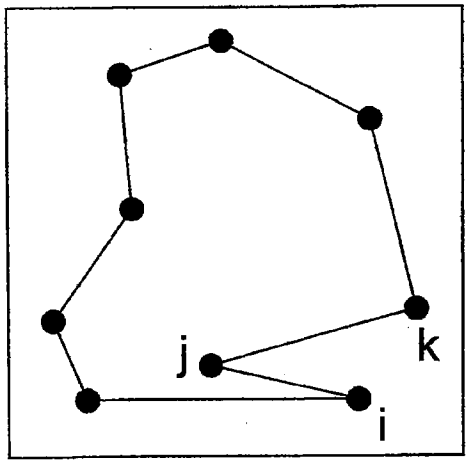
- A recent article compares several improvement procedures on 30 Euclidean TSPs from the literature with known optimal solutions. These problems range in size from 105 to 2392 nodes. In each case, the initial tour is constructed using a "nearest neighbor" heuristic. The best of the arc-exchange procedures, Iterated Lin-Kernighan, is the most computationally burdensome, but it consistently obtains results that are within 1% of optimality.

Quality of tour improvement procedures

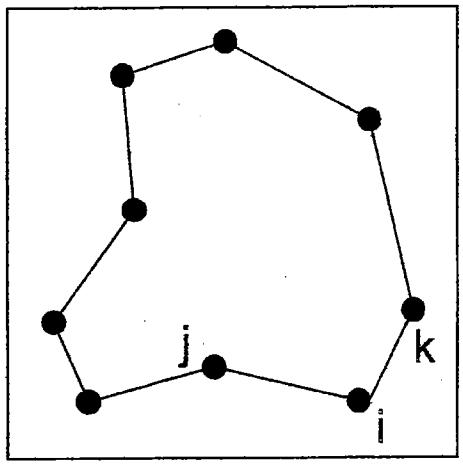
Heuristic	Average percent above optimality
Two-opt	8.3
Three-opt	3.8
Lin-Kernighan (Variant 1)	1.9
Lin-Kernighan (Variant 2)	1.5
Iterated Lin-Kernighan	0.6

COMPOSITE PROCEDURES

- In the figure below, we illustrate the concepts of tour construction and tour improvement. In particular, we point out that the hypothetical tour construction sequence of i-j-k in (a) is improved to obtain j-i-k in (b).



(a)



(b)

- When we apply a tour construction procedure followed by a tour improvement procedure, we have a composite procedure.

A Powerful Convex Hull Based Heuristic

Step 1. Form the convex hull of the set of nodes. The hull gives the initial subtour.

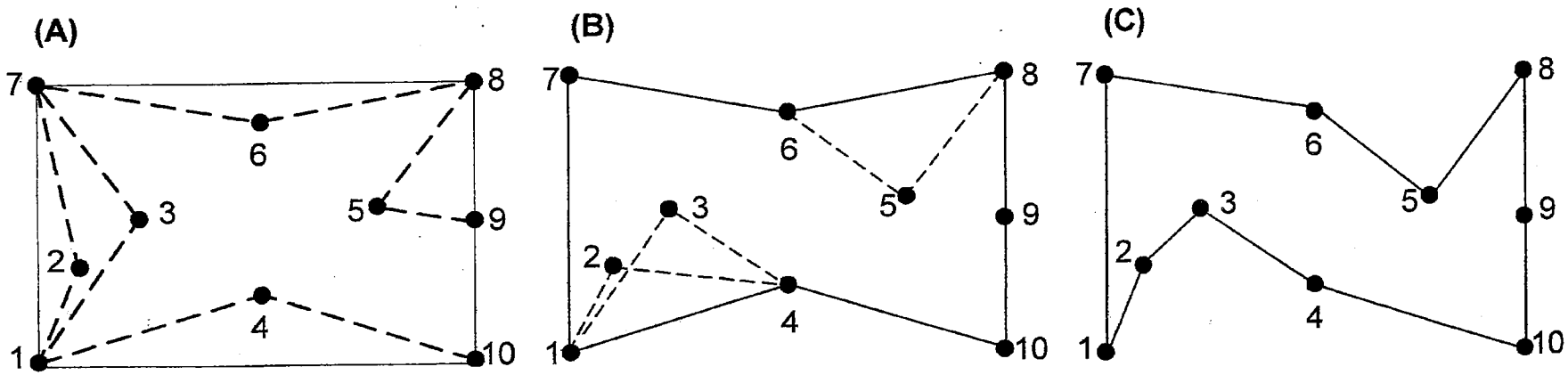
Step 2. For each node k not yet contained in the subtour, identify the two adjacent nodes i_k and j_k on the subtour such that $c_{i_k k} + c_{k j_k} - c_{i_k j_k}$ is minimal.

Step 3. Insert the node k* that maximizes the angle between arcs (i_{k^*}, k^*) and (k^*, j_{k^*}) in the subtour between i_{k^*} and j_{k^*}.

Step 4. Repeat Steps 2 and 3 until a Hamiltonian cycle is obtained.

Step 5. Apply the OROPT procedure to the tour generated in Steps 1-4. When no further improvements are available, stop.

EXAMPLE OF CONVEX-HULL BASED HEURISTIC



(A) Initial subtour and insertions.

(B) Intermediate subtour and insertions.

(C) Final tour.

OTHER INTERESTING HEURISTICS

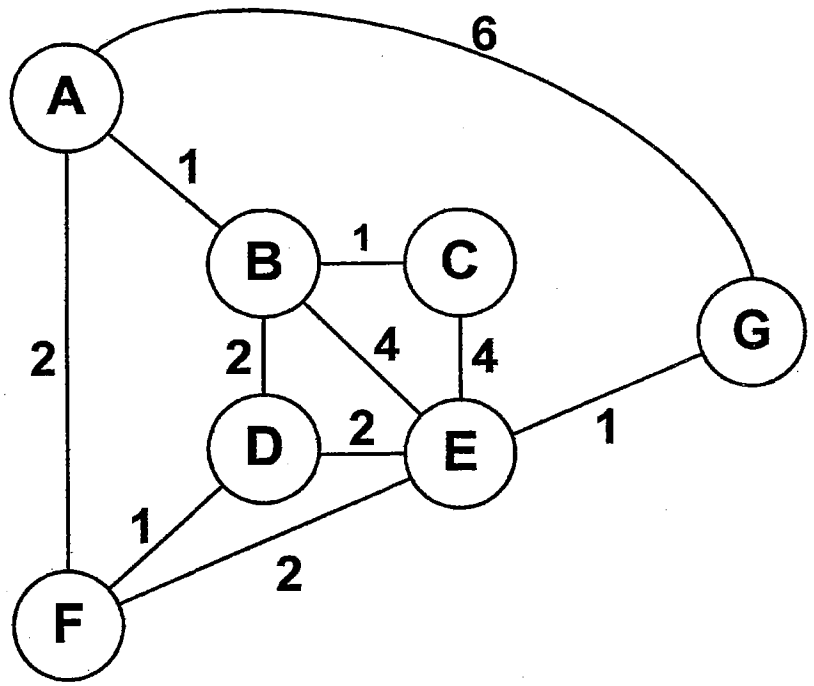
- Karp's partitioning heuristic
- Simulated annealing, Tabu search
- Genetic algorithms
- Elastic nets (neural networks)

RELATED ISSUES

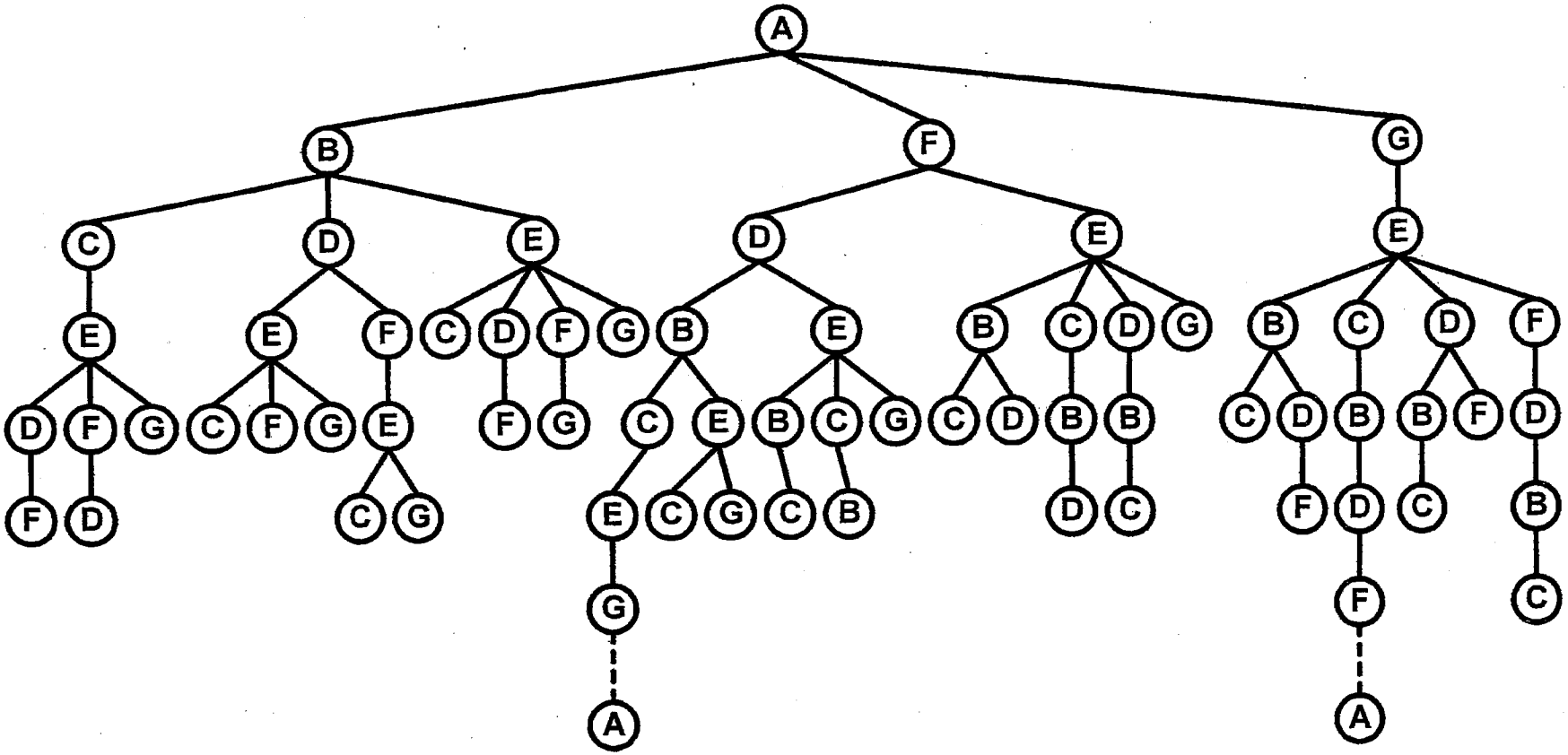
- Asymmetric TSPs
- Orienteering variants
- Arc routing problems

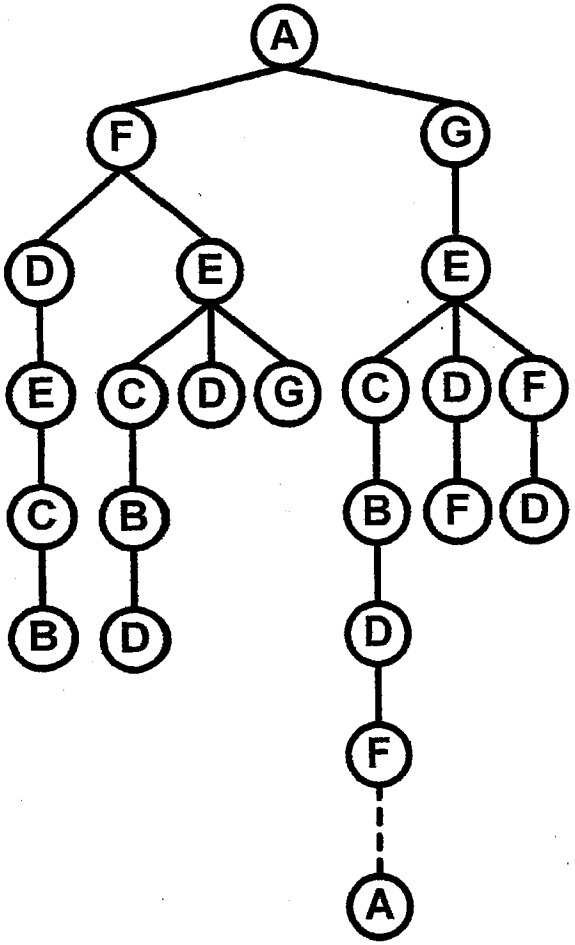
BACKTRACKING FOR SOLVING TSPs OPTIMALLY

A SMALL TSP EXAMPLE



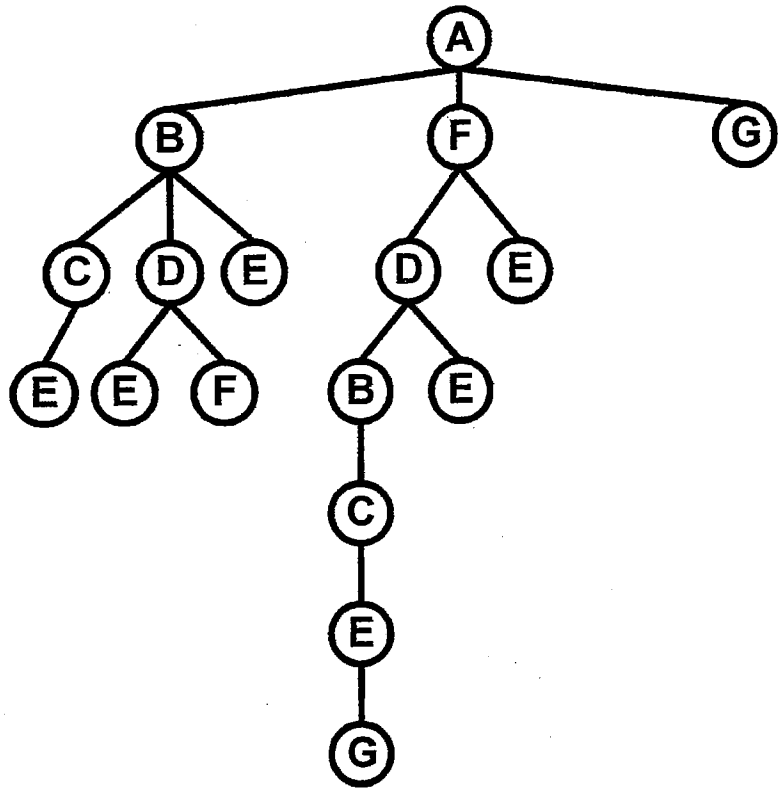
EXHAUSTIVE SEARCH WITH DEPTH-FIRST SEARCH





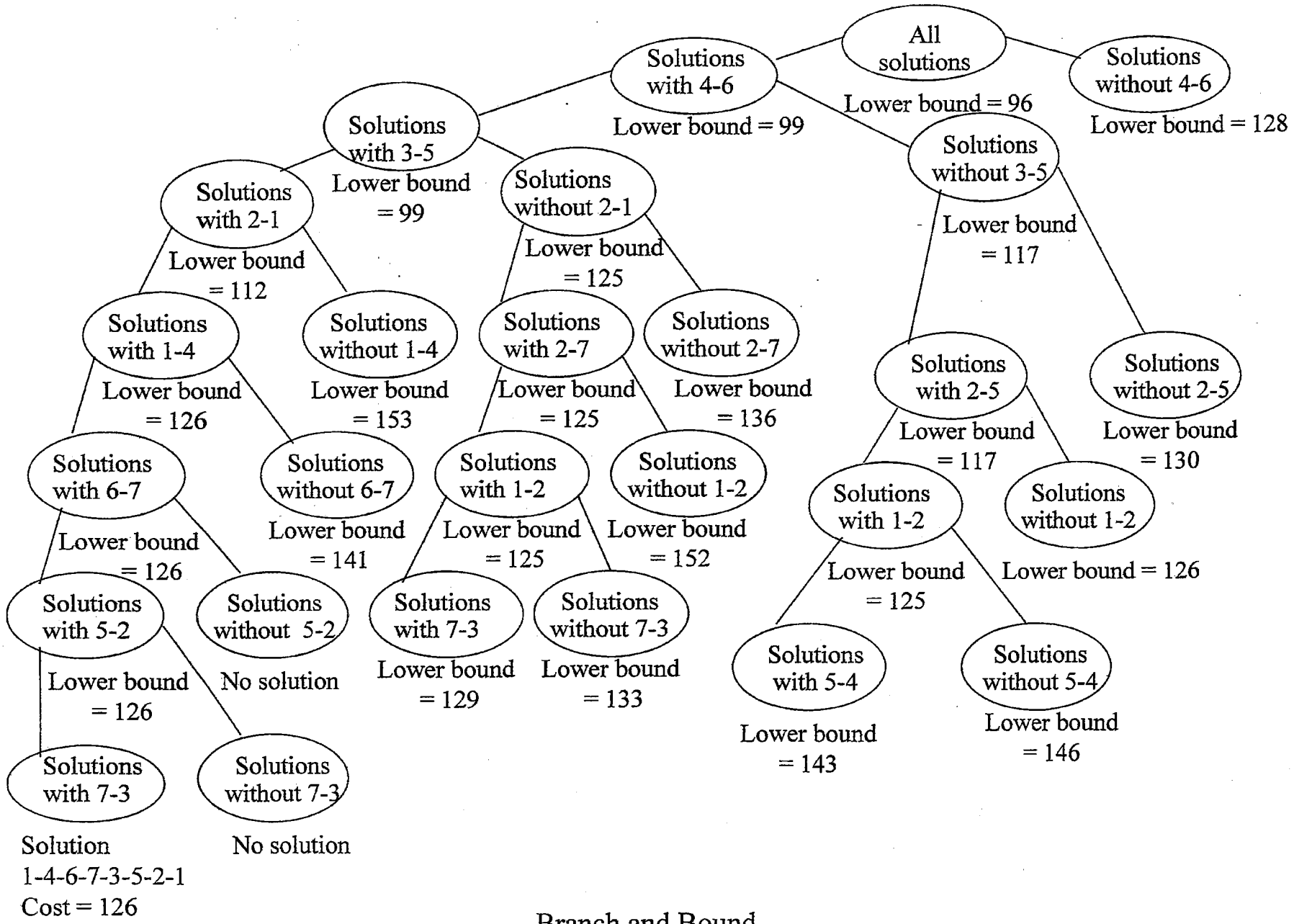
PRUNING THE TREE BY REMOVING SYMMETRIES

- Insist upon B after C after A.



PRUNING THE TREE BASED UPON COST

- Exploit connectedness and bounds.

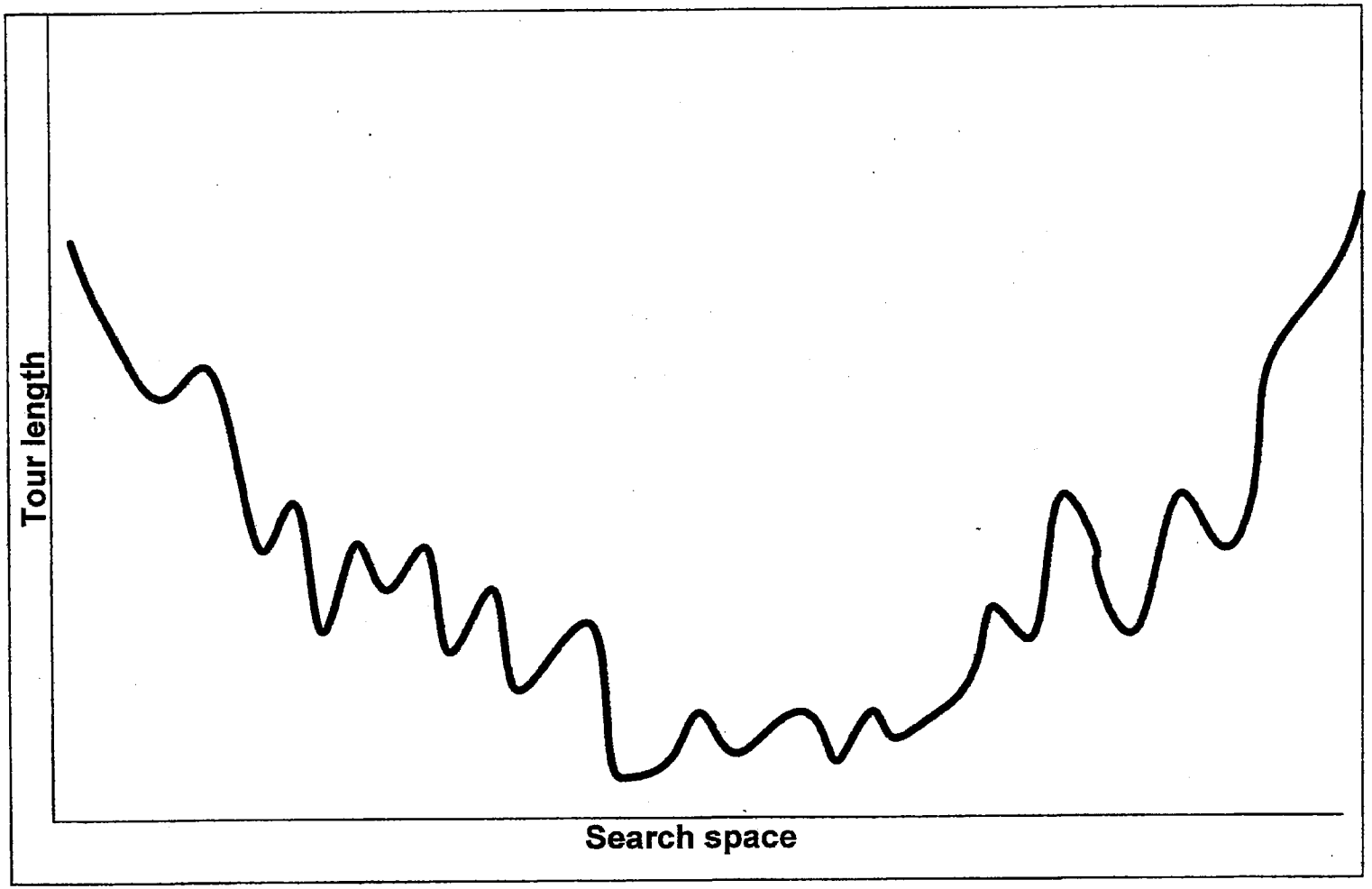


Branch and Bound

METAHEURISTICS

- Simulated Annealing
- Deterministic Threshold Accepting
- Deterministic Record-to-Record Travel
- Deterministic Great Deluge
- Tabu Search
- Genetic Algorithms
- Neural Networks

MOTIVATION BEHIND METAHEURISTICS



SIMULATED ANNEALING

Step 1. Compute an initial Hamiltonian tour T and choose an initial temperature τ and a repetition factor r.

Step 2. As long as the stopping criterion is not satisfied perform the sub-steps.

a. Do the following r times.

(i) Perform a random modification of the current tour to obtain the tour T′ and let $\Delta = c(T') - c(T)$, the difference in tour lengths.

(ii) If $\Delta < 0$ (improvement), then set $T = T'$. Otherwise set $T = T'$ with probability $= e^{-\Delta/\tau}$.

b. Update τ and r.

Step 3. Output the best solution found.

DETERMINISTIC THRESHOLD ACCEPTING ALGORITHM

Calculate initial OBJ value from initial Hamiltonian tour.

Set initial threshold value T

Set old OBJ = initial OBJ

For threshold values T, . . ., 0

    For all pairwise exchanges

        Calculate new OBJ

        If(new OBJ < old OBJ + T)

            Use this tour as current

            Set old OBJ = new OBJ

DETERMINISTIC RECORD-TO-RECORD TRAVEL

```
Calculate initial RECORD value from initial Hamiltonian tour
Set old RECORD = initial RECORD + 1
Set fixed DEVIATION
While ( RECORD < old RECORD )
    old RECORD  = RECORD
    For all pairwise exchanges
        Calculate new OBJ
        If( new OBJ < RECORD + DEVIATION )
                Use this tour as current
                If ( OBJ < RECORD ) Then
                    Set RECORD = OBJ and store record tour
```

TABU SEARCH

Step 1. Compute an initial Hamiltonian tour T and start with an empty tabu list $\mathcal{L}$.

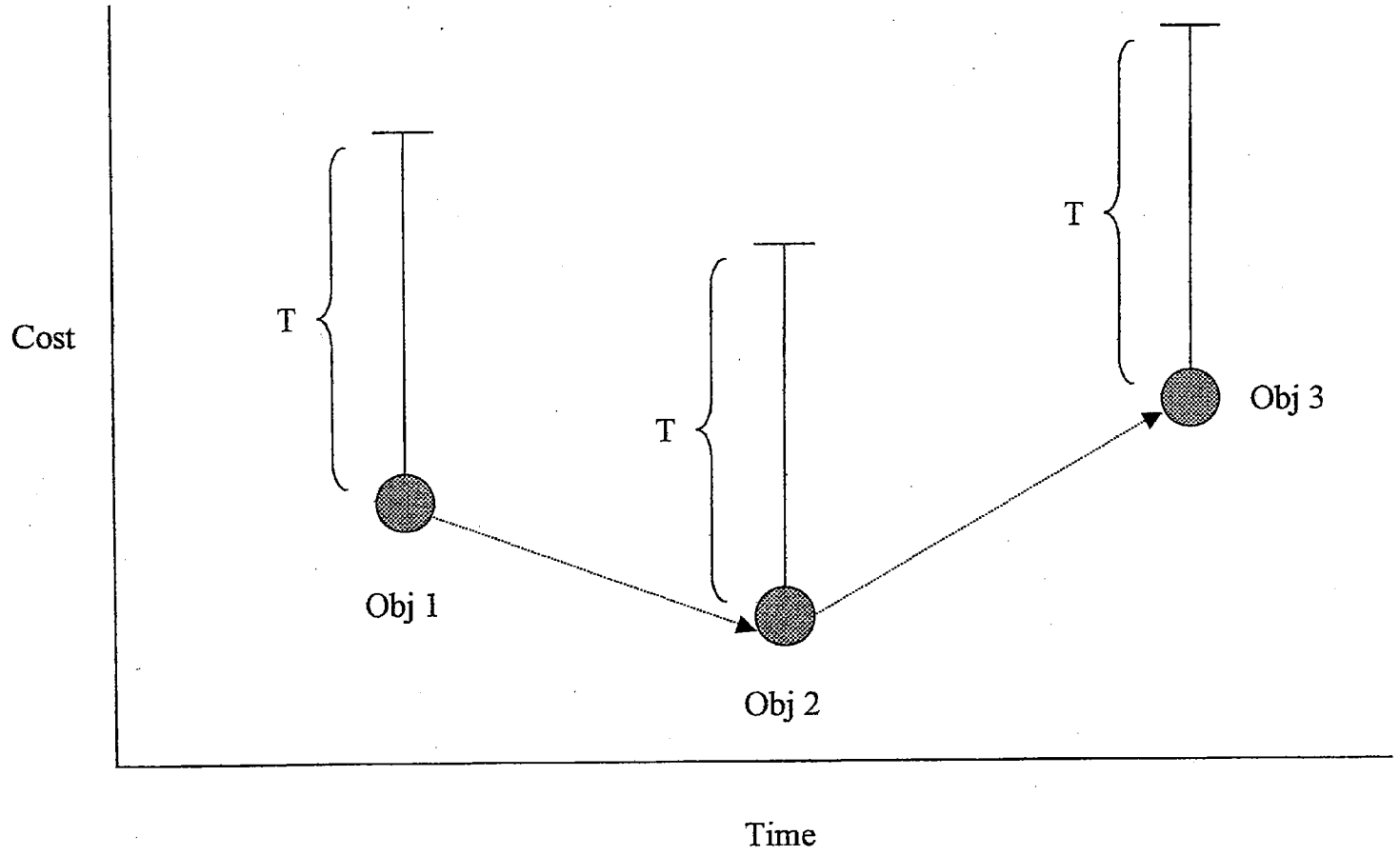
Step 2. As long as the stopping criterion is not satisfied, perform the following sub-steps.

a. Perform the best move that is not forbidden by $\mathcal{L}$.

b. Update the tabu list $\mathcal{L}$.

Step 3. Output the best solution found.

Threshold Accept



Record to Record Travel