Neural and Evolutionary Computing.

Lab 3: Combinatorial Optimization Problems. Simulated Annealing

---

1. TSP – Traveling Salesman Problem

TSP is a well known combinatorial optimization problem asking to find the optimal route for a salesman who has to visit a set of n towns. It is a constrained optimization problem characterized by:

- Constraint: the salesman visits each town exactly once
- Objective function: the cost of the tour (should be minimized)

The classical TSP is equivalent with the problem of finding an optimal Hamiltonian tour in a complete graph. TSP can be solved exactly for small values of n but, since the number of possible tours is (n-1)!/2, for large values of n there are no efficient exact methods. TSP belongs to the class of NP-complete problems.

TSP is important not only from a theoretical point of view but also from a practical point of view since there are several real-world problems which can be formulated as a TSP:

- Vehicle Routing Problem (VRP): find the optimal route for vehicles
- Control of drilling machines which are used to construct boards for integrated circuits
- Find shortest routes through selections of airports in the world
- Reconstruct DNA sequences starting from subsequences

Other applications are listed at [http://www.tsp.gatech.edu/apps/index.html]

TSP has several variants:

- *Asymmetric TSP*: the passing from one node to another one depends on the tour orientation.
- *Sequential Ordering Problem* – SOP: there are additional constraints specifying that a given node should be visited before another one.
- *Capacitated vehicle routing problem* – CVRP: find optimal tours for a set of trucks which have to transport products from a warehouse to different customers. The trucks have all the same capacity.
- *Generalized TSP*: the nodes correspond to clusters of locations and there are several arcs between nodes.

Besides exact methods there exist a lot of heuristic methods based on incremental improvements of the current tour. One of the most used heuristics for TSP is the Lin-Kernighan heuristic which is based on replacing some arcs of the current tour with other ones such that the total cost becomes smaller. The simplest case is when just two arcs are replaced (2-opt transformation) which is equivalent to reversing the order of visiting the nodes belonging to a subtour.

Example: Let us consiuder 6 nodes: A,B,C,D,E,F . If the current tour is (A,C,B,E,F,D), by replacing the arc (A,C) with the arc (A,F), and the arc (F,D) with (C,D) and by reversing the order of visiting the nodes B and E one obtains the tour (A,F,E,B,C,D). It is easy to see that this transformation can be obtained by reversing the subtour (C,B,E,F).

2. Solving TSP using Simulated Annealing

In order to solve a problem by using Simulated Annealing there are several elements to be chosen:

- *Solution encoding.* The natural encoding variant for TSP is the permutation: a tour through n nodes can be described as a permutation of order n. This encoding ensures the satisfaction of the constraint of visiting only once each node.

  Example: If the towns are numbered as follows: 1-A, 2-B, 3-C, 4-D, 5-E, 6-F then the route (A,C,B,E,F,D) corresponds to the permutation (1,3,2,5,6,4)

- *Local search mechanism (construction of a new configuration starting from the existing one)* The simplest mechanism is based on a 2-opt transformation:
  - Choose two random indices i and j such that 1<=i<j<=n
  - Reverse the order of elements in the permutation having indices between i and j.

  Example: If the current tour is described by the permutation (1,3,2,5,6,4) and i=2, j=5 then the new permutation will be: (1,6,5,2,3,4)

- *Acceptance probability.* The probability to accept a configuration C' obtained from the configuration C can be computed by using the Boltzmann distribution:

  P(C'|C)=min{1,exp(-(cost(C')-cost(C))/T(k)}

  where cost(C) is the cost of configuration C and T(k) is a control parameter (temperature).

- *Cooling schedule.* If T(k) denotes the temperature corresponding to the iteration k then the value corresponding to the next iteration can be computed as follows:

  - *T(k+1)=T(0)/log(k+c),* c being a constant
  - *T(k+1)=T(0)/k*
  - *T(k+1)=a T(k),* with *a* denoting a value smaller but close to 1 (e.g. a=0.99)

Remark. The initial value of the temperature (T(0)) should be large enough to allow the transition between any two configurations.

- *Stopping condition.* The stopping criterion can be related to the value of the temperature (stop when the temperature is low enough), to the number of iterations (stop after a given number of iterations) or to the value of the objective function (cost of the tour).

Application 2. Implement the Simulated Annealing for TSP. Use test instances from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Hint. A variant is implemented in SA_TSP.m.

Call: SA_TSP(10000,0.001)

Exercises:

1. Write a Scilab function to read data from *.tsp files (downloaded from TSPLib) and analyze the behavior of the algorithm for the problems: eil51.tsp, eil76.tsp, eil101.tsp
2. Test the algorithm for each of the cooling schedules mentioned above.
3. Modify the function which compute the cost of a tour such that the distance between two nodes is computed only once (hint: store the distances in a matrix)

Homework: Implement a SA algorithm to solve the discrete knapsack problem (for a knapsack of capacity C and a set of n objects of sizes $d_1,d_2,\ldots,d_n$ and values $v_1,v_2,\ldots.,v_n$ find a subset of objects which fit into the knapsack and have a maximal value).

Hint: use a binary representation for a solution (a candidate solution x will be a vector of n binary elements where the value on position i is 1 if object i is selected and 0 if object i is not selected). In order to deal with the constraint that the total size of the selected objects is less than C you can use the penalty method by adding to the total value a term which penalizes the cases when the constraint is not satisfied.