ANNALS OF FACULTY ENGINEERING HUNEDOARA
– INTERNATIONAL JOURNAL OF ENGINEERING
Tome X (Year 2012) – FASCICULE 3 (ISSN 1584 – 2673)

[1.] Anca Elena IORDAN

A COMPARATIVE STUDY OF META-HEURISTICS METHODS FOR TRAVELLING SALESMAN PROBLEM

[1.] UNIVERSITY "POLITEHNICA" OF TIMISOARA, FACULTY ENGINEERING OF HUNEDOARA, ROMANIA

ABSTRACT: *The paper presents a simulation study of the usefulness of a number of meta-heuristics used as optimization method for traveling salesman problem. The three considered approaches are outlined: Neighborhood Search, Hill Climbing and Simulated Annealing. Using a purpose-developed computer program, efficiency of the meta-heuristics has been studied and compared. The modeling of the environment is achieved through specific UML diagrams representing the stages of analysis, design and implementation. Implementation of informatics system is realized in Java programming language.*
KEYWORDS: *TSP, Neighborhood Search, Hill Climbing, Simulated Annealing, Java*

TRAVELING SALESMAN PROBLEM

The traveling salesman problem is a well known optimization problem. Optimal solutions to small instances can be found in reasonable time by linear programming. However, since the TSP is NP-hard, it will be very time consuming to solve larger instances with guaranteed optimality. Setting optimality aside, there's a bunch of algorithms offering comparably fast running time and still yielding near optimal solutions.

The traveling salesman problem is to find the shortest Hamiltonian cycle [1] in a graph. This problem is NP-hard and thus interesting. There are a number of algorithms used to find optimal tours, but none are feasible for large instances since they all grow exponentially.

In traveling salesman problem, we have a set of N cities C={C_1, C_2, …, C_n} and a set, E, of routes connecting the cities with one another. In other words, we have a full connected graph G(C, E), where C is the set of nodes and E={E_{ij}}, i=1,2,…,N, j=1,2,…,N, i≠j, is a set of edges. Each edge E_{ij} (connecting nodes C_i and C_j) has a measure d_{ij} – the distance between cities C_i and C_j. the task is to find the shortest route between all the cities, assuming that each city has to be visited exactly once.

In mathematical terms, TSP is defined as a problem of finding the minimal-length Hamiltonian circuit is the sum of distances $d_{i,j}$ of all edges $E_{i,j}$ belonging to the tour.

META-HEURISTICS METHODS – Neighborhood Search Method

We are given an instance I of a combinatorial optimization problem where X is the set of feasible solutions for the instance (we write X(I) when we need to emphasize the connection between instance and solution set) and c : X -> R is a function that maps from a solution to its cost. X is assumed to be finite, but is usually an extremely large set. We assume that the combinatorial optimization problem is a minimization problem, that is, we want to find a solution x such that c(x*) ≤ c(x), $\forall x \in X$.*

We define a neighborhood of a solution $x \in X$ as $N(x) \subseteq X$. That is, N is a function that maps a solution to a set of solutions. A solution x is said to be locally optimal or a local optimum with respect to a neighborhood N if c(x) ≤ c(x'), $\forall x' \in N(x)$. With these definitions it is possible to define a neighborhood search method [2].

The algorithm takes an initial solution x as input. It computes $x' = \arg\min_{x'' \in N(x)} \{c(x'')\}$, that is, it finds the cheapest solution x' in the neighborhood of x. If c(x') < c(x) then the method performs the update x = x'. The neighborhood of the new solution x is searched for an improving solution and this is repeated until a local optimum x is reached. When this happens the method stops. The algorithm is denoted a steepest descent method as it always chooses the best solution in the neighborhood.

A simple example of a neighborhood for the TSP is the 2-opt neighborhood which can be traced back to. The neighborhood of a solution x in the 2-opt neighborhood is the set of solutions that can be reached from x by deleting two edges in x and adding two other edges in order to reconnect the tour.

Hill Climbing

Hill Climbing [3] can be described as a method to find a solution of a problem which is, like the name imply, hill climbing. To be more precise, when you climb a hill, you will always go from lower ground to higher ground, not other wise, just like that, a hill climbing will always try to find a better alternative to reach a solution. In HC, the method will generate several states from an initial state, and then it will analyze all the alternatives, and find the best one. After that, it will move to that state, the

same thing happen to that state, until either the solution is found or there're no better alternatives that is generated (in this case, the current state is assumed as the solution).

Simulated Annealing

A hill-climbing algorithm that never makes "downhill" moves towards states with lower value (or higher cost) is guaranteed to be incomplete, because it can get stuck on a local maximum [4]. In contrast, a purely random walk-that is, moving to a successor chosen uniformly at random from the set of successors-is complete, but extremely inefficient. Therefore, it seems reasonable to try to combine hill climbing with a random walk in some way that yields both Simulated Annealing efficiency and completeness. Simulated annealing is such a method.

DEVELOPMENT STAGES OF THE INFORMATICS SYSTEM – System's analysis

In order to compare the obtained results for Neighborhood Search, Hill Climbing and Simulated Annealing methods, there was implemented an interactive Java application. For object orientated development of the application, unified modeling language will be used. To achieve UML diagrams was used the ArgoUML [5] software.

The analysis of an informatics system consists in drawing the use case and activity diagrams [6]. The informatics system will be described in a clear and concise manner by representation of the use-cases. Each case describes the interaction between the user and the system. The diagram defines the system's domain, allowing visualization of the size and scope of the whole developing process. This diagram includes:

- *One actor - the user who is external entity with which the didactic game interacts.*
- *Nine use cases that describe the functionality of the interactive game.*
- *Relationships between user and use cases (association relationships), and relationships between use cases (dependency and generalization relationships).*

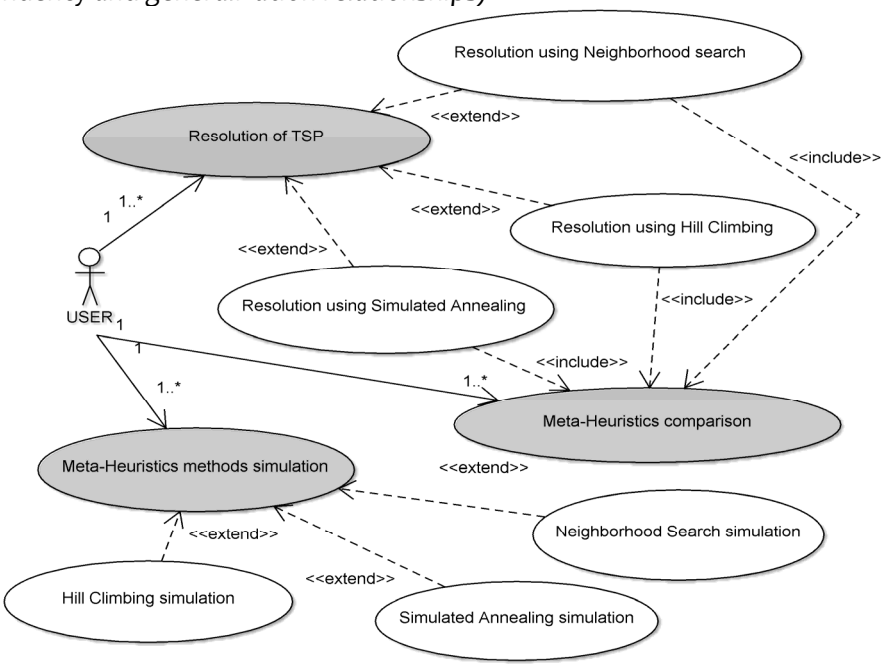

Figure 1. Use Cases Diagram

System's designing

Conceptual modeling allows identifying the most important concepts for the interactive software. Class diagram [7] is represented in figure 2 in order to be observed the connection mode between the classes and the interfaces that are used and also the composition and aggregate relationships between instances.

For memorizing the vertices of the graph has been implemented "Vertex" class. For memorizing the edges of the graph has been implemented "Edge" class. For memorizing a graph has been implemented "Graph" class. For memorizing a state configuration, there was implemented "TSPState" class which implements "State" interface. For memorizing a node of search tree, there was implemented "TSPNode" class which realizes "Node" interface and for memorizing an expanded node together with its successors, there was implemented "TSPExpandedNode" class which realizes "ExpandedNode" interface.

Hill Climbing method is implemented by using "TSPHillClimbing" class which inherit from abstract class "TSPMetaheuirstic". Neighborhood Search method is implemented by using "TSPNeighborhoodSearchMethod" class which inherit from abstract class "TSPMetaheuirstic".

Simulated Annealing method is implemented by using "TSPSimulatedAnnealing" class which inherits from abstract class "TSPMetaheuirstic". For realizing the window that will compose the graphical interface of the application and for simulation of Neighborhood Search, Hill Climbing and Simulated Annealing methods, there were implemented the "TSP" class.

USER INTERFACE

The informatics system is accomplished using the Java programming language [8]. The application can easily convert in a Java applet.

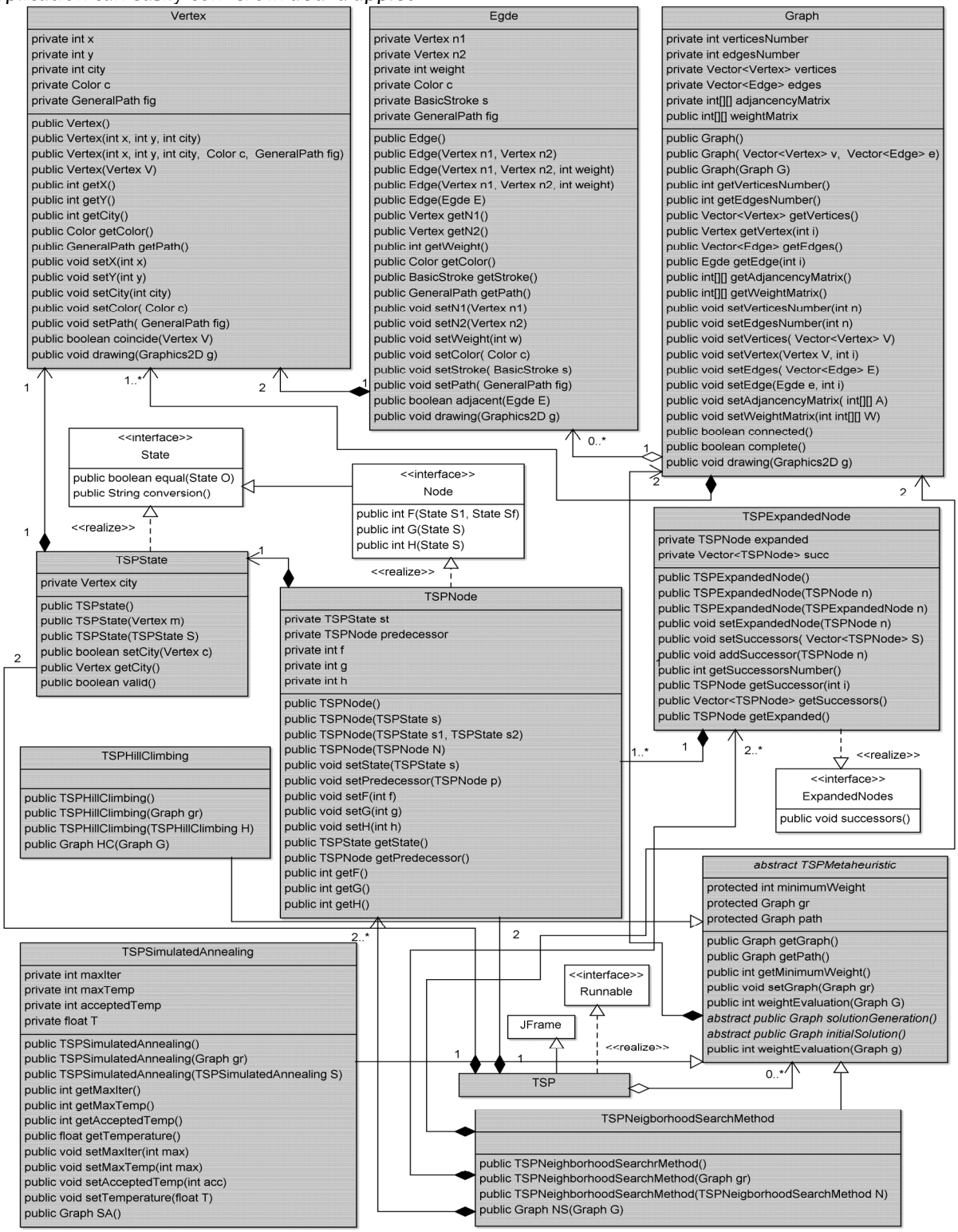

Figure 2. Class Diagram

Starting from specified requisites in uses cases diagram (figure 1) it was designed graphical user interface of the informatics system. The main page of the application contains buttons for selecting the following options: Neighborhood Search simulation (figure 3), Hill Climbing simulation and Simulated Annealing simulation.

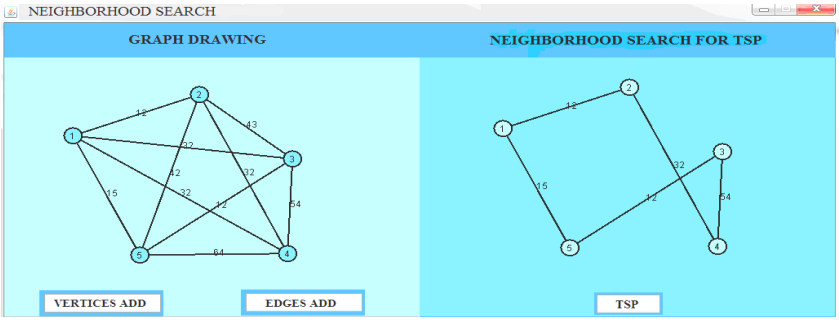

Figure 3. Neigborhood Search for TSP with five cities

EXPERIMENTALS RESULTS

Because one of the objectives of this paper is to compare the efficiency of selected methods, the presented results have been obtained using each method with the most suitable parameters. In this experiment the methods are used for a complete TSP, which means that all the cities are directly connected. Simulations for TSP consisting of different number of cities, varying from 15 to 20, have been made. Results obtained are collected in table 1 and in figure 4. All the used meta-heuristic methods are able to find an approaching route of optimal route for a TSP, but Simulated Annealing is the most efficient meta-heuristic methods.

Table 1. Results given by different methods for TSP

Number of cities	Meta-heuristics methods		
	NS	HC	SA
15	341	364	338
16	214	223	223
17	245	251	273
18	264	274	272
19	300	357	349
20	320	330	330

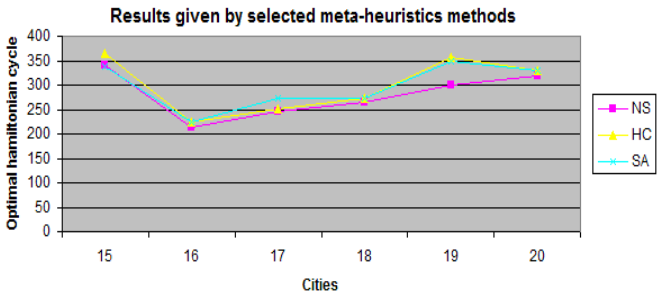

Figure 4. Graphic of experimental results

CONCLUSIONS

All meta-heuristics tested in the paper are very popular optimization techniques. Three approaches have been selected and simulation studies of their efficiency on the basis on one task, the traveling salesman problem, have been performed. All experiments show that the best results, from the point of view of space complexity, as well of time complexity, are produced by Simulated Annealing.

The theme treated in this paper is of very actually, discipline Artificial Intelligence being an important component in the student formation which studies in the Computer Science domain, but even in domains related to it.

REFERENCES

[1.] *S. Russel, P. Norving, Artificial intelligence – A modern approach, Prentice Hall, 2003*
[2.] *K. Ahuja, B. James, P. Abraham, A survey of very large-scale neighbourhood search techniques, Discrete Applied Mathematics 123, 75–102, 2002*
[3.] *J. Riera-Ledesma, J. Salazar-Gonzalez, A Heuristic approach for the Travelling Purchaser Problem, European Journal of Operational Research 162, 142–152, 2005*
[4.] *C. Rego, F. Glover, Local Search and Metaheuristics for the Travelling Salesman Problem, G. Gutin and A.P. Punnen (eds.), The Travelling Salesman Problem and its Variations., 2002*
[5.] *http://argouml.tigris.org*
[6.] *G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison Wesley, 1999*
[7.] *M. Fowler, K. Scott, UML Distilled: A Brief Guide to the Standard Object Modeling Language, Addison Wesley, Readings MA, USA, 2000*
[8.] *S. Tănasă, C. Olaru, S. Andrei, Java de la 0 la expert, Polirom Press, Iasi, 2007*

ANNALS OF FACULTY ENGINEERING HUNEDOARA
– INTERNATIONAL JOURNAL OF ENGINEERING