STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 2, 2004

HEURISTICS AND LEARNING APPROACHES FOR SOLVING THE TRAVELING SALESMAN PROBLEM

GABRIELA ŞERBAN AND CAMELIA-MIHAELA PINTEA

Abstract. In present, all known algorithms for NP-complete problems are requiring time that is exponential in the problem size. Heuristics are a way to improve time for determining an exact or approximate solution for NP-complete problems.

In this article is introduced and solved a problem based on a generalization of the Traveling Salesman Problem. We compare two classical algorithm results for the application: Branch and Bound and Nearest Neighbor and also two Ant Algorithms: Ant System and Ant Colony System. Being stochastic algorithms, Ant Algorithms have the solutions chosen according to a probability, which depends on the pheromone level, therefore they can be also considered as reinforcement learning techniques.

We also propose a reinforcement Q-learning method for solving the Traveling Salesman Problem.

Keywords: Combinatorial Optimization, Traveling Salesman Problem, Symmetric Graphs, Branch and Bound, Nearest Neighbor, Ant Algorithms, Heuristics, Agents, Reinforcement Learning.

1. Introduction

One of the most important and promising research field in recent years has been Heuristics from Nature. These heuristics, utilizing analogies with natural or social systems are using to derive non-deterministic heuristic methods and obtain very good results in NP-hard combinatorial optimization problems.

A heuristic *is a commonsense rule (or set of rules) intended to increase the probability of solving some problems. It is a general formulation that serves to guide investigation.*

Let us consider the Traveling Salesman Problem. The Traveling Salesman Problem is one of the most studied discrete optimization problems. TSP has many variations and a large number of applications.

Received by the editors: December 10, 2004.

2000 *Mathematics Subject Classification.* 68T05, 68T20, 90C27.

1998 *CR Categories and Descriptors.* I.2.6 [**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.2.8 [**Computing Methodologies**]: Artificial Intelligence – *Problem Solving, Control Methods, and Search*; G.1.6 [**Mathematics of Computing**]: Numerical Analysis – *Optimization* .

Today problems are harder and harder to solve, because of the multitude of inputs and the time needed to produce best results. That is why, in such cases we try to find an approximation of the best solution, using a heuristic.

A **construction heuristic** is an algorithm that determines a tour, in the graph associated to the problem, according to some construction rules, but does not try any improvements upon this tour. **Improvement heuristics** are characterized by a certain type of basic move to alter the current tour. We can combine these two types of heuristics in the following way: first we use a constructive heuristic to construct a tour. Then we repeatedly use an improvement heuristic to improve the current tour.

Heuristics in Artificial Intelligence are obtained using a certain amount of repeated trials, employing one or more agents, neurons, particles, chromosomes, ants. Two main features have to be balanced: the degree of exploits and the degree of exploration.

In case of multiple individuals, a cooperation or competition among individuals take place. These agents improve the solution quality at each iteration of the process and are able to adapt to new situations. So, these heuristics are adaptive (to the changes in the environment).

The most used construction heuristic is Nearest Neighbor Heuristic, which has gained in popularity because of the TS problem. The salesman starts at some city and then visits the city nearest to the starting city. From there he visits the nearest city that was not visited so far and so on until all cities will be visited, and the salesman returns to the starting point. It is probably close to the real salesman's approach. But is a poor heuristic, however.

Other heuristics often applied are the **genetic algorithms**, the **neural networks** and the heuristics inspired by social insects: **swarm intelligence**, **ant algorithms**.

In section 2, the subsection Theoretical support, we introduce a problem based on a generalization of Symmetric Traveling Salesman Problem. We will compare the optimal results obtained with two algorithms, in the classical approach, Branch and Bound and Nearest Neighbor and the results are shown in a table.

In the section Ant Algorithms we compare the results of two algorithms Ant System and Ant Colony System, which are applied for the problem mentioned above. The later one is more efficient and faster as we will see from the tables. There are also comparative graphics, that illustrate better the results.

Learning techniques can be considered as a kind of metaheuristics, that is why we consider in section 3 a reinforcement learning approach for solving the TSP problem.

2. A generalization of the Symmetric Traveling Salesman Problem

2.1. **Theoretical support.** As is shown in [1], these are the definitions used for Generalized Symmetric Travel Salesman Problems.

The Generalized Symmetric Traveling Salesman Problem (GSTSP):

Definition 2.1 *Given a weighted complete digraph K_N and a partition $V_1, V_2, \ldots, V_k$ of vertices, find a minimum weight cycle containing exactly one vertex from each set $V_i, i = 1, \ldots, k$.*

Definition 2.2 *The sets V_i, are called* **clusters** *and a cycle containing exactly one vertex from each cluster is called a* **tour**.

A common known definition of the Traveling Salesman Problem:

Definition 2.3 *Given N cities, if a salesman starting from his home city his goal is to visit each city exactly once and then return home, find the order of a tour such that the total distance travelled is minimum.*

We introduce the following **problem**:

Given N cities, if a salesman is starting from his home city and his house and want to visit each city exactly once, through exactly one house, and then return home, find the order of a tour such that the total distance travelled is minimum.

In this work, we consider that every city has a given number of houses. The salesman crosses each city once going trough one house from a city. We know the distances between houses from each city. We want to know the shortest way from the starting house and city making a tour of all cities, coming back in the same house and city.

The salesman chooses each time the house that is nearest from an unvisited city in the Nearest Neighbor method, and in the Branch and Bound method he visits all the cities and a random houses from each city, if there is a way between them.

The input files are:

- travel.in including a matrix N x N with 0 and 1, the matrix of distances ex. : (0,1,1,0,1), (1,0,1,1,1), (1,1,0,1,1), (0,1,1,0,1), (1,1,1,1,0);
- travel.txt specifies the city numbers and the numbers of houses in each city ex. (5,1)(1,2)(2,3)(3,1)(4,3)(5,1);
- distance.in specifies the distances between houses of different cities in a $N \times 5$ matrix.

File distance.in

1, 1, 2, 1, 5	1, 1, 5, 1, 15	3, 1, 5, 1, 6	1, 1, 2, 2, 17
1, 2, 5, 1, 1	2, 1, 4, 1, 55	1, 1, 2, 3, 55	3, 1, 4, 1, 90
1, 2, 2, 1, 60	3, 1, 4, 2, 6	2, 3, 4, 1, 43	1, 2, 2, 2, 8
3, 1, 4, 3, 58	2, 1, 4, 2, 20	1, 2, 2, 3, 44	2, 1, 5, 1, 2
2, 2, 4, 2, 21	1, 1, 3, 1, 33	2, 2, 5, 1, 21	2, 3, 4, 2, 24
1, 2, 3, 1, 4	2, 3, 5, 1, 1	2, 1, 4, 3, 15	2, 1, 3, 1, 54
4, 1, 5, 1, 12	2, 2, 4, 3, 7	2, 2, 3, 1, 31	4, 2, 5, 1, 11
2, 3, 4, 3, 9	2, 3, 3, 1, 2	4, 3, 5, 1, 13	

2.2. **Classical approach.** We use two methods for solving this application, namely Branch and Bound where we randomly choose a house from a city and Nearest Neighbor technique. For Branch and Bound we make 500 runs for each starting house from each city. In some cases Nearest Neighbor failed because there is not a way between cities 1-4.

We denote BB the Branch and Bound method, and NN the Nearest Neighbor method.

The results are presented in Table 1.

City	House	NN	NN Tours	BB	BB Tours
1	1,2		*Failed*	35	(1,1)(5,1)(4,2)(3,1)(2,3)
2	1	33	(2,1)(5,1)(1,2)(3,1)(4,2)	21	(1,2)(5,1)(4,2)(3,1)(2,3)
2	2	56	(2,2)(4,3)(5,1)(1,2)(3,1)	24	(2,1)(5,1)(4,2)(3,1)(1,2)
2	3	36	(2,3)(5,1)(1,2)(3,1)(4,2)	21	(2,3)(3,1)(4,2)(5,1)(1,2)
3	1		*Failed*	15	(3,1)(1,2)(5,1)(2,3)(4,3)
4	1	62	(4,1)(5,1)(1,2)(3,1)(2,3)	20	(4,1)(5,1)(1,2)(3,1)(2,3)
4	2		*Failed*	13	(4,2)(3,1)(1,2)(5,1)(2,3)
4	3	80	(4,3)(2,2)(1,2)(5,1)(3,1)	16	(4,3)(2,3)(3,1)(1,2)(5,1)
5	1	29	(5,1)(1,2)(3,1)(2,3)(4,3)	29	(5,1)(1,2)(3,1)(2,3)(4,3)

Table 1.

Remark 2.1 *From Table 1 we see that Branch and Bound found the best minimum solution and in the last case the solutions are equal, where the computational complexity is low.*

We mention that the time complexity of the NN approach is $O(N^2)$.

Comparative results are also illustrated in Figure 1.

2.3. **Ant Algorithms.** This section shows a heuristic from nature, Ant Algorithm [10]. First, this natural heuristic was introduced in [11], [12].

The Ant Systems have proved to be efficient and robust global optimization methods. Ant Colony Algorithms are very complex hybrid systems. Inspired from the nature, they combine the power of metaheuristic in a highly distributed way. It can be considered as an evolutionary method, where agents are interacting in a cooperative multi-agent system. Ant systems, like Genetic Algorithms, are population based approaches.

At all ant-based algorithms the basic idea is the positive feedback mechanism. A virtual pheromone, used as reinforcement, allows good solutions to be kept in memory. It is also important to avoid good, but not very good solutions from becoming reinforced, which can lead to premature convergence, also called stagnation.

In order to escape local optima traps, a so-called negative feedback is used through pheromone evaporation.

Cooperative behavior is another important concept: ant colony algorithms make use of the simultaneous exploration of different solutions. Ant colonies are multi-agent systems, where the ants have the role of the cooperative agents. The communications between agents is done indirectly using the above-mentioned pheromones, which is also mentioned as the concept of stimergy.

The Ant Colony System Colony was developed to fill the gaps of the AS Algorithm, making more efficient and robust. The aim was to improve the performance of AS, that was able to find good solutions within a reasonable time only for small size problems.

ACS is based on several modifications of AS: a different transition rule, a different pheromone trail update rule, the use of local updates of pheromone trail to favor exploration.

The results for Ant System (AS) and Ant Colony System (ACS) are presented in Table 2.

City	House	AS	AS Tours	ACS	ACS Tours
1	1	57	(1,1)(2,1)(5,1)(4,2)(3,1)	57	(1,1)(2,1)(5,1)(4,2)(3,1)
1	2	38	(1,2)(2,2)(4,3)(5,1)(3,1)	38	(1,2)(2,2)(4,3)(5,1)(3,1)
2	1	91	(2,1)(1,1)(5,1)(4,2)(3,1)	33	(2,1)(5,1)(1,2)(3,1)(4,2)
2	2	56	(2,2)(4,3)(5,1)(1,2)(3,1)	56	(2,2)(4,3)(5,1)(1,2)(3,1)
2	3	29	(2,3)(4,3)(5,1)(1,2)(3,1)	29	(2,3)(4,3)(5,1)(1,2)(3,1)
3	1	29	(3,1)(2,3)(4,3)(5,1)(1,2)	29	(3,1)(2,3)(4,3)(5,1)(1,2)
4	1	184	(4,1)(2,3)(1,2)(5,1)(3,1)	105	(4,1)(5,1)(1,1)(3,1)(2,3)
4	2	33	(4,2)(2,1)(5,1)(1,2)(3,1)	36	(4,2)(2,3)(5,1)(1,2)(3,1)
4	3	80	(4,3)(2,2)(1,2)(5,1)(3,1)	73	(4,3)(2,3)(5,1)(1,2)(3,1)
5	1	75	(5,1)(4,2)(2,1)(1,1)(3,1)	57	(5,1)(2,1)(1,1)(3,1)(4,2)

Table 2.

Remark 2.2 *In half of the cases ACS-Algorithm had better results then AS-Algorithm.*

Comparative results are shown in Figure 2.

We have to mention that adding more agents (ants) to an Ant System leads to better performance.

The complexity of one cycle of the Ant Algorithm is $O(N^2 \cdot M)$ (N is the number of inputs, M is the number of ants), so it takes very long time to solve the problem with many inputs. Researchers have wondered how many artificial ants (agents) are needed for a given problem and they have found that the number of agents must be approximately equal to the number of inputs [12].

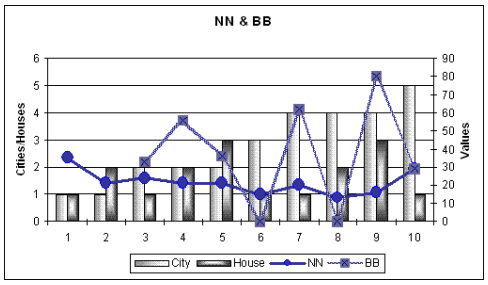

FIGURE 1. NN-BB

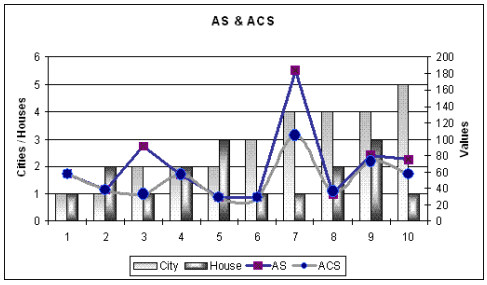

FIGURE 2. AS-ACS

3. Learning approach for solving the TSP problem

In this section we propose a reinforcement learning approach for solving the TSP problem.

Let us consider a finite number of "cities" and their coordinates on a map. We suppose that the cost of travel between a pair of cities is given by the euclidian between them. An agent (the traveling salesman) has to find the cheapest way of visiting all the cities and to return to the starting point. So, the goal of the agent is to learn by reinforcement the optimal policy.

In Reinforcement learning [14], the agent is simply given a goal to achieve and then the agent learns how to achieve that goal by trial-and-error interactions with its environment.

In a reinforcement learning problem, the agent receives a feedback, known as *reward* or *reinforcement*; the reward is received at the end, in a terminal state, or in any other state, where the agent has correct information about what he did well or wrong.

3.1. The SARSA Algorithm.

SARSA [13] is a reinforcement Q-learning algorithm, which combines the advantages of Temporal difference learning and Monte Carlo learning methods.

From the TD methods, the algorithm takes the advantage of learning at each step, instead of waiting the end of an episode. From the Monte Carlo methods, SARSA takes the advantage of going back and using the rewards obtained in each state in order to update the values of the previous action-state pairs and the capacity of functioning without the model of the environment in which the learning takes place [16].

The idea of the SARSA algorithm [13] is to apply the Temporal Difference methods to the state-action pairs, in comparison with the classical methods, where this methods are applied only to the states.

SARSA converges with probability 1 to an optimal policy and action-value function as long as all state-action pairs are visited an infinite number of times and the policy converges in the limit to the Greedy policy.

The agent learns the optimal policy after some training sequences. A training sequence consists of a number of episodes, during which the agent has the following behavior:

- until a final state is reached (the final state is given by the goal of the agent) or until the number of steps exceed a maximum number, using an action selection mechanism (ϵ-Greedy or SoftMax [13]), the agent updates properly the Q-values (the values of the state-action pairs).

The complete description of the SARSA algorithm is given in [17].

3.2. **Experimental results.** In order to apply the SARSA algorithm described above, we make the following assumptions:

- there are N cities that the agent has to visit;
- from a given city, the agent can visit any other city (the graph is complete);
- the agent starts from the city 1 (the initial state in the RL task);
- the states in the RL task are the cities the agent can visit;
- a city reached by the agent at a given moment is considered to be a final state if the agent has visited all the cities so far;
- the possible actions of the agent in a given state (city) are *N-1* (the other cities that the agent can visit, excepting the current state);
- as an action selection mechanism we use the ϵ-Greedy approach;
- the reward received by the agent in a given state (city) is considered to be the euclidian distance between the current city and the city numbered with 1.

Because of the probabilistic action selection mechanism during the learning process, we repeat the training sequence of the agent several times.

For a simplest evaluation of the results, we choose the example in which there are 4 cities. The coordinates of the cities are given in Table 3.

City	Coordinates on the map
1	(10,10)
2	(10,11)
3	(11,10)
4	(13,11)

Table 3. Coordinates of the cities on the map.

We mention that the optimal tour is 1,2,4,3,1 and the total distance of this tour is **7,24**. The NN approach determines an approximation of the solution, 1,3,2,4,1 with the total distance **8,57**.

The results obtained after applying our learning algorithm are shown in Table 4.

Training sequence	Number of episodes	Tour	Total Distance
1	6	1,4,2,3,1	**8,57**
2	6	1,3,2,4,1	**8,57**
3	6	1,2,4,3,1	**7,24**
4	6	1,2,3,4,1	**7,81**
5	6	1,3,2,4,1	**8,57**
6	6	1,4,3,2,1	**7,81**
7	6	1,4,3,2,1	**7,81**
8	6	1,2,4,3,1	**7,24**

Table 4. Results obtained with the SARSA algorithm.

The results obtained with SARSA are better than the result obtained with the NN approach.

We have applied the algorithm for different values of **N** and for different number of training episodes and we observe that SARSA, in average, gives better results than the NN approach.

We have to mention that the time complexity of the TSP solving with the RL approach during one training sequence is O(N*e), where e is the number of training episodes.

3.3. **Further work.** Further work can be done in the following directions:

- how to improve the learning rate;
- to apply other learning approaches for solving TSP (like LRTA* algorithm [18]);
- the generalization of the above approach for multi agent systems (systems in which several agents has to cooperatively learn by reinforcement to solve the TSP problem).

4. Conclusions

Any combinatorial optimization problem has an associated graph, which means that these problems can be solved in terms of graph theory and also provides us with an easy way to view the problem.

An heuristic method reduces the time complexity of the problem solving. As an example, a brute force approach for TSP has $O(2^N)$ complexity (N is the number of cities).

A dynamic programming algorithm also gives the global optimum for this problem (and for many other hard combinatorial optimization problems) but also takes exponential time.

Section 2 is based on GSTSP, where we can compare the cities with clusters and a house with a vertex. A cycle containing exact one house - vertex- from each city- cluster- is a tour. Branch and Bound is a consistent technique that exhaustive explore a searching space, following at each step all the minimum possibilities.

The Nearest Neighbor method we will apply in the case when we want an approximate solution, but obtained in a short time, with a reduced computational complexity. We can improve methods by taking randomly the equal minimal values, at each new execution of the algorithm.

We can extend the problem requiring that the traveller should cross exactly trough a given number of houses in each city, where the given number could be the minimum values of the houses from the cities.

In section 2 the same problem is solved with Ant Algorithms. Ant Algorithms are primarily used for combinatorial optimization, especially for NP-hard tasks. The Traveling Salesman Problem (TSP) is an important application, for which were developed several Ant Colony methods. The basic idea of Ant System is that of simulating the behavior of a set of agents that cooperate to solve an optimization problem by means of simple communications.

As a disadvantage it can be stated the multiple parameters used for these algorithms. It is very hard to set up the optimal values of the parameters, which can be very different from one problem to another.

As a conclusion, we may say that even though classic algorithms give an **exact** solution, heuristics may give very good solution, or even the best one and are very **handy techniques** because they are usually cheap to apply and combine. We should also try to combine heuristics with exact algorithms. Different heuristics can be obtained as follows: using a Greedy algorithm to choose each move, using multiple agents that start from different points in the graph, using clustering techniques to group regions in the search space, using clustering techniques to group agents, using local search techniques to improve solutions.

References

[1] G. Gutin, *Traveling Salesman and Related Problems*,Royal Holloway, Univ.of London, 2003

[2] G. Gutin, A. Punnen, *The Traveling Salesman Problem and its variations*,Kluwer Academic Publishers

[3] M. Dorigo, L.M. Gambardella, *Ant Colonies for the Traveling Salesman Problem*, BioSystems 43, 73-81, 1997

[4] M. Dorigo, L.M. Gambardella, *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*, IEEE Transactions on Evolutionary Computation 1(1) 53-66, 1997

[5] M. Dorigo, V. Maniezzo, A. Colorni, *The Ant System: Optimization by a Colony of Cooperating Agents*, IEEE Transaction on Systems, Man, and Cybernetics-Part B, 26(1) 29-41, 1996

[6] M. Dorigo, G. Di Caro, *The Ant Colony Optimization Metaheuristic*, in D. Corne, M. Dorigo, F. Glover(Eds.), New Ideas in Optimization, McGraw-Hill, 1999

[7] E. Bonabeau, M. Dorigo, G. Therraulaz, *Swarm intelligence: from natural to artificial systems*, Oxford University Press, 1999

[8] M. Hozefa Botee, E. Bonabeau, *Evolving Ant Colony Optimization*, Advances in Complex Systems 1 (2/3) 149-159, 1998

[9] V. Maniezzo, A. Carbonaro, *Ant Colony Optimization: An Overview*, in Celso C. Ribeiro, Pierre Hansen (Eds.), Essays an Surveys in Metaheuristics, Kluwer Academic Publishers, Boston/ Dordrecht/ London, pp. 21-44, 2002

[10] A. Colorni, F. Dorigo, V. Maniezzo, G. Righini, M. Trubian, *Heuristics from nature for hard combinatorial optimization problems* Published in International Transactions in Operational Research, 3,1, pag.1-21

[11] A. Colorni, F. Dorigo, V. Maniezzo, *Distributed Optimization by Ant Colonies* Proceedings of ECAL-91-European Conference on Artificial Life, Paris, France, F. Varela and P. Bourgine (Eds.), Elsevier Publishing, 1991,pp. 134-142

[12] A. Colorni, F. Dorigo, V. Maniezzo, *An Investigation of Some Properties of an Ant Algorithm* Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92), Brussels, Belgium, R. Manner, B. Manderick(Eds.), Elsevier Publishing, 1992, pp. 509-520.

[13] R. Sutton, A.G. Barto, *Reinforcement learning*, The MIT Press, Cambridge, England, 1998

[14] S.J.Russell, P.Norvig, *Artificial intelligence. A modern approach*, Prentice-Hall International, 1995

[15] M. Harmon, S. Harmon, *Reinforcement Learning - A Tutorial*, Wright State University, www-anw.cs.umass.edu/*sim*mharmon/rltutorial/frames. html, 2000

[16] A. Perez-Uribe, E. Sanchez, *Blackjack as a TestBed for Learning strategies in Neural Networks*, Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'98

[17] G. Serban, *A Reinforcement Learning Intelligent Agent*, Studia Universitatis "Babes-Bolyai", Informatica, XLVI, Number 2, pp.9-18, 2001

[18] G. Weiss, *Multiagent systems - A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, Cambridge, Massachusetts, London, 1999

"Babeş-Bolyai" University
E-mail address: gabis@cs.ubbcluj.ro

"George Coşbuc" National College
E-mail address: cami_mihaela@yahoo.com