Journal of Theoretical and Applied Information Technology
31st July 2012. Vol. 41 No.2
© 2005 - 2012 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A MULTI-AGENTS SYSTEM ARCHITECTURE TO RESOLVE AN NP-COMPLETE PROBLEM

[1]ABDOUN OTMAN, [2]ABOUCHABAKA JAAFAR AND [3]TAJANI CHAKIR
Faculty of sciences IbnTofail University, Kenitra Morocco
Email: [1]otman.fsk@gmail.com , [2]aboucha06-univ@yahoo.fr , [3]chakir_tajani@hotmail.fr

ABSTRACT

The multi-agent system (MAS) is a nature-inspired method, which supports cooperative search by the self-organization of a group of compact agents situated in an environment with certain sharing public knowledge. Moreover, each agent in MAS is an autonomous entity with personal declarative memory and behavioral components. In recent years, multi-agent systems have become a more and more important in the field of artificial intelligence and specifically in complex systems. In this paper, MAS is refined for solving the traveling salesman problem (TSP), which is a classic hard computational problem.

Keywords: *Artificial Intelligent, Multi-agent Systems, NP-Complete Problem, Travelling Salesman Problem*

1. INTRODUCTION

Problems that can be solved by algorithms in polynomial time are considered tube so called easy problems. For a problem of size n the time needed to find absolution is a polynomial function of n. Harder problems requires on the other hand an exponential function of n, which of course means that the execution time grows much faster than for an easy problem, when the size of the problem increases [1], [12].NP-complete problems are hard problems to solve. They belong to a class of computational problems, for which no deterministic polynomial algorithm has been found. The list of NP-complete is long, there exits several thousands problems. They are represented within many different areas as graph theory, network, scheduling, games and puzzles etc…

The Travelling Salesman problem (TSP) is a well known NP-complete problem [2]. It is the problem of finding the least-cost round-trip route that visits a number of cities exactly once and then returns to the starting city. The given information is the cities and the costs of travelling from any city to any other city. In the M-TSP the m-salesman has to cover the given cities and each city must be visited by exactly one salesman. Every salesman starts from the same city, called depot, and must return at the end of his journey to this city again.

The Traveling Salesman Problem (TSP) [3]-[4] is a classic combinatorial optimization problem. Although it can be easily formulated, it exhibits various interesting aspects of hard computational problems and has often served as a touchstone for novel approaches [5]-[6].

Moreover, TSP has various applications, such as very large scale integration (VLSI) design [10], rearrangement clustering [8], predicting protein functions [9], etc…

Meta-heuristics have proven useful when solving NP-complete problems. It is therefore natural to explore the possibility that meta-heuristics are also suitable for solving Traveling Salesman Problem, such as Ant Colony Optimization [10] and Genetic Algorithms [1]-[3].

The multi-agent systems (MAS) have potential advantages in solving problems associated with open systems, distributed and complex. The multi-agent system is a nature-inspired method, which addresses the self-organization of agents working with limited declarative knowledge and simple procedural knowledge under ecological rationality [11]. Specifically, agents explore in parallel based on socially biased individual learning and indirectly interact with other agents through sharing public information organized in the environment.

In this paper, MAS is refined to implement simple and efficient knowledge components for solving TSP. In Section II, some existing knowledge components for TSP, particularly search behaviors and related auxiliary data structures, are described. In Section III, some Resolution Methods of TSP are presented. In Section IV, the Multi-Agents approach to resolve a TSP is described. In the last section, this paper is concluded.

2. THE TRAVELING SALESMAN PROBLEM

The Traveling Salesman Problem (TSP) is one of the most intensively studied problems in computational mathematics. In a practical form, the problem is that of a traveling salesman who has to tour a certain number of cities, leaving from one of them, going exactly once through the others, to return at the end to the city from which he departed [1]. This traveler wants to minimize the total distance covered. How should he plan his itinerary for minimum total cost of the entire tour?

The search space for the TSP is a set of permutations of *n* cities. Any single permutation of n cities yields a solution (which is a complete tour of n cities). The optimal solution is a permutation which yields the minimum cost of the tour. The size of the search space is *n!* [1].

In other words, a TSP of size *V* is defined by a set of points $v = \{v_1, v_2, \ldots, v_n\}$ which v_i a city marked by coordinates $v_{i.x}$ and $v_{i.y}$ where we define a metric distance function *f* (1). A solution of TSP problem is a form of scheduling *T = (T[1], T[2],……, T[n], T[1])* which *T[i]* is a permutation on the set *{1, 2, …, V}.*

The evaluation function calculates the adaptation of each solution of the problem by the following formula:

$$f = \sum_{i=1}^{n-1} \sqrt{(v_i.x - v_{i+1}.x)^2 + (v_i.y - v_{i+1}.y)^2} + \sqrt{(v_n.x - v_1.x)^2 + (v_n.y - v_1.y)^2} \quad (1)$$

Where n is the number of cities.

If d, a distance matrix, is added to the TSP problem, and *d(i, j)* a distance between the city v_i and v_j (2), so the cost function *f* (1) is expressed as follows:

$$\mathrm{d(i\,,j)} = \sqrt{\left(v_i.x - v_j.x\right)^2 + \left(v_i.y - v_j.y\right)^2} \quad (2)$$

$$f(T) = \sum_{i=1}^{n-1} \mathrm{d(T[i], T[i+1])} + \mathrm{d(T[n], T[1])} \quad (3)$$

The mathematical formulation of TSP problem expresses by:

$$min\{f(T), T = (T[1], T[2], \ldots\ldots, T[n])\} \quad (4)$$

Which T[i] is a permutation on the set {1, 2, …,V}.

The travelling salesman problem (TSP) is an NP-hard problem in combinatorial optimization studied in operations research and theoretical computer science [3].

A quick calculation shows that the complexity is *O(n!)* which n is the number of cities (Table. 1) [1] and [2].

Table 1. Number of possibilities and calculation time by the number of cities

Number of cities	Number of possibilities	Computation time
5	12	12 µs
10	181440	0,18 ms
15	43 billions	12 hours
20	60 E+15	1928 years
25	310 E+21	9,8 billions of years

3. RESOLUTION METHODS OF TSP

3.1. Ant Colony

We describe an artificial ant colony capable of solving the traveling salesman problem (TSP). Ants of the artificial colony are able to generate successively shorter feasible tours by using information accumulated in the form of a pheromone trail deposited on the edges of the TSP graph. Computer simulations demonstrate that the artificial ant colony is capable of generating good solutions to both symmetric and asymmetric instances of the TSP [16].

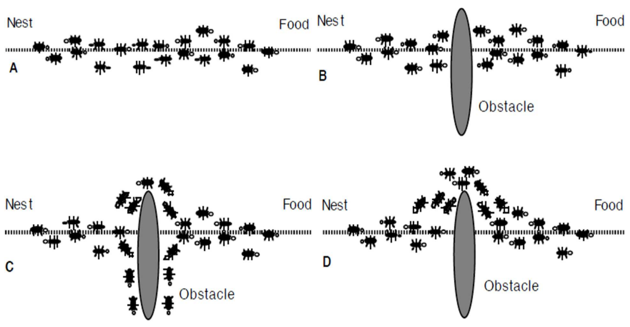

Figure 1. (A) Real ants follow a path between nest and food source. (B) An obstacle appears on the path: Ants choose whether to turn left or right with equal probability. (C) Pheromone is deposited more quickly on the shorter path. (D) All ants have chosen the shorter path.

In recent years, many research works have been devoted to ant colony optimization (ACO) techniques indifferent areas. It is a relatively novel meta-heuristic technique and has been successfully used in many applications especially problems in combinatorial optimization. ACO algorithm models the behavior of real ant colonies in establishing the shortest path between food sources and nests. Ants can communicate with one another through chemicals called pheromones in their immediate environment. The ants release pheromone on the ground while walking from their nest to food and then go back to the nest. The ants move according to the amount of pheromones, the richer the pheromone trail on a path is, the more likely it would-be followed by other ants.

So a shorter path has a higher amount of pheromone in probability, ants will tend to choose a shorter path. Through this mechanism, ants will eventually find the shortest path. Artificial ants imitate the behavior of real ants, but can solve much more complicated problem than real ants can [17].

ACO has been widely applied to solving various combinatorial optimization problems such as Traveling Salesman Problem (TSP) [9], [15], [16], Job-shop Scheduling Problem (JSP), Vehicle Routing Problem (VRP), Quadratic Assignment Problem (QAP), etc.

Ants are a biological example of a self-organized system. Travelling between their nest and food sources, they deposit a chemical pheromone, guiding themselves towards optimal routes between nest and food source (figure 1). It so happens that the chemical pheromone is detectable by the ants, and that they prefer to follow paths with large amounts of pheromone deposited. This way the ants will follow the shortest route – more pheromone will be deposited on the shortest path, as more ants will be able to cover this distance in the same time as the time taken to travel any other path.

In mathematical terms they form a minimum spanning tree of the Euclidean graph with vertices defined by their nest and food sources. Though complex algorithms exist for computing minimum spanning trees, the ants are only controlled by local interactions [14].

M. Dorigo proposed in his Ph.D thesis to use this ant behavior as a way to solve the TSP [16] :

- A number of artificial ants (not necessarily |V| ants) moves from vertex to vertex in the graph, depositing pheromone on edges travelled. The starting vertex of each ant is chosen at random.
- Ants probabilistically prefer edges with a lot of pheromone deposited and/or light edges, but they are required not to revisit vertices already visited.
- When travelling an edge the ant performs a local trail update. A local trail update adds pheromone to the edge, but it also subtracts a certain amount to image the evaporation of chemical pheromone in nature.
- When all ants have completed a cycle of the graph, the ant that followed the lightest cycle of the graph, performs a global trail update: It modifies the pheromone amount of each edge in its cycle by adding an amount that is inversely proportional to the weight of the cycle.

This process is iterated until some stopping criterion (usually that no large improvement to the cycle weights occurs for some number of consecutive iterations). While the local trail update serves to direct the behavior of the ants towards some level of diversification, by evaporating some of the deposited pheromone, the global trail update serves to intensify the search near the solutions.

But, the Ant Colony method has an inconvenient we cite:

- Although ACO has a powerful capacity to find out solutions to combinational optimization problems, it has the problems of stagnation and premature convergence and the convergence speed of ACO is very slow. Those problems will be more obvious when the problem size increases.
- The generic ant colony optimization algorithm for the TSP does not handle incomplete graphs very well. It is a requirement that each ant can traverse a Hamiltonian cycle of the graph, no matter which edges it chooses on its way. This requirement however cannot be satisfied by an incomplete graph.

3.2. Genetic Algorithm

Compared to evolutionary biology, instead of studying populations of creatures, solutions to combinatory optimization problems are the subjects in the populations of genetic algorithms (GA).As with local search an evaluation function is used to determine the fitness of the solutions [1]. A standard GA for a combinatory optimization problem has the form:

1. Initialize the population with individuals being representations of solutions.
2. Genetic Operators: Reproduction, Mutation and Elimination.
3. Return best solution from the population.

Compared to the classical optimization algorithms, the genetic algorithm has several advantages as:

- Use only the evaluation of the objective function regardless of its nature. In fact we do not require any special property of the function to be optimized (continuity, differentiability, connectedness, ..), which gives it more flexibility and a wide range of applications ;
- Generation has a parallel form by working on several points at once (population of size N) instead of a single iteration in the classical algorithms ;
- The use of probabilistic transition rules (crossover and mutation probability), as opposed to deterministic algorithms where the transition between two individuals is required by the structure and nature of the algorithm [1].

The reproduction function is often referred to as a crossover function, due to the origin in biology where a chromosomal crossover is the exchange of material between two chromosomes.
The following, very frequent used scheme was used in [1-3]:

- The representation for a TSP solution is a list of vertices – completely equivalent to the mathematical representation of the cycle.
- The evaluation function is then just a mapping from a cycle to its weight (the sum of the weights of all its edges).

3.3. Multi-Agents System

It is assumed that the reader has some knowledge about agents and multi-agent systems (MAS), and therefore this section will only give a brief description of agents and multi-agent systems. Since a multi-agent system is a system consisting of individual agents, its necessary first to define what an agent is. The following definition is given in [15]:

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

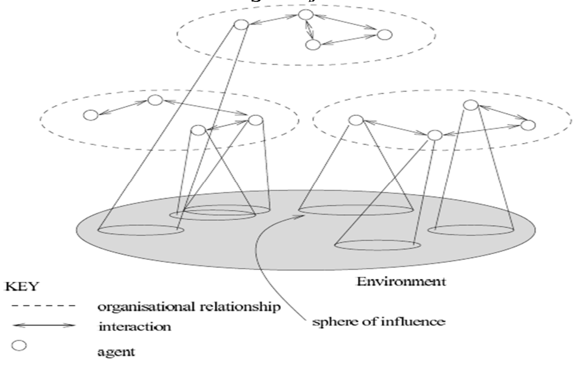

Figure 2. Typical structure of MAS. Figure from [17]

A MAS is a composed system of several agents which interact and work together in order to archive certain goals [18]. Their interactions can be either co-operative or selfish. That is, the agents can share a common goal, or they can pursue their own interests. The typical characteristics of MAS's are that each agent has incomplete information or capabilities for solving the problem.

I.e. each agent can have a local perception of the global state and need to co-operate in an autonomous and asynchronous way with other agents in order to meet the goals of the global system.

To get a more illustratively presentation of a MAS, see figure 2. Here is the typical structure of a MAS shown. The system contains multiple agents who interact through a communication protocol, which are indicated with the double pointing arrows. The agents are able to act in the environment (grey sphere), but with different influence on the environment.

The spheres of influence show the different parts of the environment the agents have influence over. These spheres may coincide in some cases, which may give rise to dependency relationships between the agents.

In [15] they give the example that two agents may both be able to move through a door, but may not be able to do so simultaneously. Additionally agents will typically be linked by other, which is show with the punctuated sphere. This could be that an agent is the boss of another.

In the following we try to analyze the common characteristics when using MAS in solving NP-complete problems. The usage of MAS when solving NP-complete problems may differ from the intuitive perception of MAS. In the following we give an overview of the different categories of multi-agent systems when solving NP-complete problems.

4. NEW COOPERATIVE APPROACH TO RESOLVE A TSP

The Cooperative approach proposed in this paper, see figure 3, to solve the Travelling Salesman Problem Contains various agents include the following:

4.1. Domain Agent (D.A.)

Domain agent which is responsible for a domain in the Travelling Salesman Problem. Its main task is to ensure that the domain constraint is satisfied.

Furthermore the domain agent has the responsibility to inform the coordinator agent about possible steps, it can take towards a solution of the TSP (solution steps).The two possible solution steps, which the agent should recognize, are:

- A cell, where the state of the domain results in a value that is unambiguous (defined as a value solution step)
- If a value has been sat in one of its cells, it should suggest to eliminate the candidate from every other cell in the domain (defined as an elimination solution step).

4.2. Naked Agent (N.A.)

The Naked Set strategy bases its elimination on knowledge about the number of candidates inside a set of cells, and the value of these candidates. Therefore when the Naked Set agent is requested to perform a search for a Naked Set, it should search through a given set of cells and determine, if they contain a Naked Set. It should not search the entire Space of location in the TSP instance at once, but instead search small parts of this space. Because the search is divided, it is possible to ensure that the agent only searches the parts of the TSP space that are relevant.

The division of the TSP Maps, space of the instance TSP locations, into domain agents comes in handy at this point, as the strategy agent can request the relevant domain agents for a list of cells, in order to avoid requesting the same domain multiple times, the agent should only request cells from domains that it has not searched before, or domains that have changed since the last time it 'visited'. Each agent can make the optimal way in its part of the TSP space.

To illustrate the search procedure executed by the Naked Agent to determine the optimal sub-path, consider the following example for five cities (Figure 4).

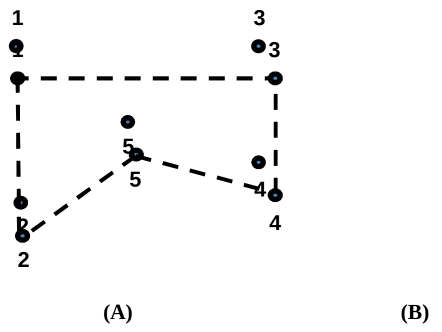

Figure 4. ***(A)*** **Initial positions for an example of 5 Cities,** ***(B)*** **The searched Optimal sub-path (5 Cities)**

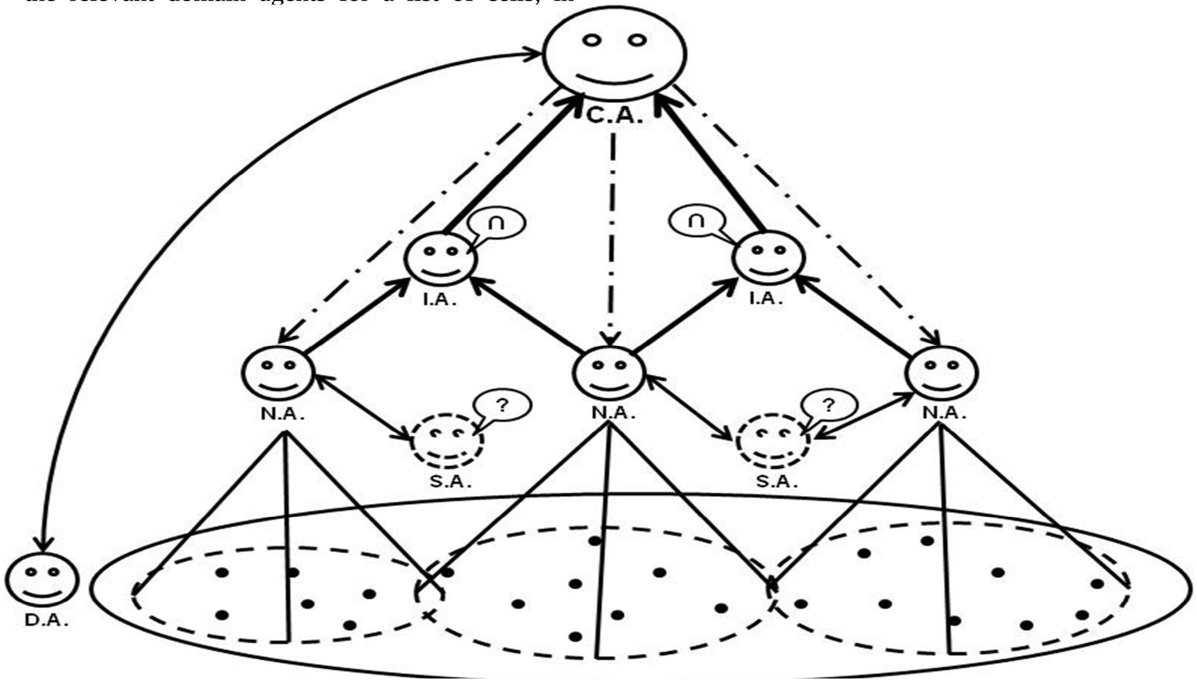

Figure 3. Schema of the proposed cooperative approach

First, the Naked Agent initializes the sub-path with a starting point “*1*” (Table 2.A). And then the agent makes calls to the function *NakedSet(1, {1},{2, 3, 4, 5}, 1, 5}*, and specifying as parameters of this function (The current city "*Stop*", cities crossed "*Neighbors*", the choices remaining "*choices*", the length of the path made up now "*lg*", the full length of the path "*max_lg*”.

At the beginning of each jump, the algorithm (figure.5) assumes that the first element of the set "*choices*" is the best, i.e. it is the nearest city "*New_neighbor*" from our stopping point "*Stop*". Then it compares its distance "*d_max*" with other distances obtained when we choose another city "*Neighbor*" of all "*choices*", provided that this latter should not be crossed before (Table 2. “B, C, E and D”).

After each iteration, and as there are still cities to cross, the Naked Agent defines the next town “*New_neighbor*”, nearest to the breakpoint current “*Stop*”, adding this city in the set of crossed cities "*neighbors*" and removing it from the set of cities to choose "*choices*".

To complete this sub-path, the traveler must return to the starting point that is why the Naked Agent specifies as last stopped the city "*1*" (Table 2.f).

Table 2. The Naked Agent strategy (or method)

1					

(A)
The sub-path takes the city"***1***"asadeparture city

1	**3**				

(B)
NakedSet(1, {1},{2,3,4,5},1,5}

1	3	**4**			

(C)
NakedSet(3,{1,3},{2,4,5},2,5}

1	3	4	**5**		

(D)
NakedSet(4,{1,3,4},{2,5},3,5}

1	3	4	5	**2**	

(E)
NakedSet(5, {1,3,4,5},{2},4,5}

1	3	4	5	2	**1**

(F)
Complete the trajectory by putting the city "***1***" as arrival city

```
Algorithm NakedSetSearch(Stop, neighbors, choices, lg, maxlg)
// Stop The current location
// neighbors the cities chosen
// choices the remaining cities
{Check if we have n stops with only n possible cities, meaning that we are at the endpoint of a sub-path of naked
cities}
iflg← count(choices) then
            return MAKE-SET(neighbors)
end if
while count(choices) > 0 do
// Search one of the towns, from the set "choices", nearest to the current city “stop”, and which has not yet been
chosen from among the towns of trajectory "neighbors"
// We assume that the first city in the set of choices “choices” is the nearest city “New_neighbor”
New_neighbor←first(choices) { Get the first cities    in choices}
//Calculate the distance between the current city “Stop” and the city recently chosen “New_neighbor “
D_min←distance(Stop, New_neighbor)
for each element of choices, lets beginning with the second choice (i=2)do
// verify, if this element includes in the set “neighbors” or not
            if include(Neighbor, neighbors) = false then
              D←distance(Neighbor, Stop)
              if D <D_min then
                     New_neighbor← Neighbor
                     D_min← D
              endif
            endif
endfor
//Add New_neighborin neighbors
choices’← choices - New_neighbor
//Remove neighbor from neighbors
neighbors’← neighbors +New_neighbor
nakedset←Naked SetSearch(New_neighbor, neighbours’,choices’, lg+1,maxlg)
end while
return MAKE-SET(empty) { Nothing was found }
```

Figure 5. Algorithm of Naked Agent strategy to find the optimal sub-path

4.3. Coordinator Agent (C.A.)

Coordinator agent which maintains the state of the TSP cell entities. It should also manage the progress of the solution, by cooperation between the domain and strategy agents. This agent should also be responsible for determining, if the TSP has been successfully solved, and should notify interested listeners that a solution was found.

4.4. Strategy Agent (S.A.)

Strategy agent which is responsible for a solution strategy heuristic. Its task is to use its strategy to suggest possible solution steps to the coordinator agent. The steps towards a solution, proposed by strategy agents, will always be elimination solution steps.

4.5. Intersection Agent (I.A.)

The intersection agent uses a slightly different strategy, than the previous two agents. In order to determine an Intersection Set, the agent needs to know the unique chains within a domain, and thereafter determine if any of the unique chains also share a second domain.

Therefore the relevant parts to search in this agent, is the square domains, since all intersection sets are also within a square domain. The agent should therefore acquire the unique chains from a relevant square domain, and determine if one of the chains also is part of a row or column domain. This can be determined fairly simple by running through the unique chains, and explore if all the cells share two common domains.

5. CONCLUSION

The Travelling Salesman Problem is not as simple a problem as it could seem, in particular it has been the basis for numerous publications and even whole books [13]. The Travelling Salesman Problem, in addition to being one of the most widely known NP-complete problems, it is as relevant for further studies as it was when Sir William Rowan Hamilton in 1856 formulated the Hamiltonian Cycle Problem – a simpler version of the Travelling Salesman Problem. In spite of the amount of work done, the Travelling Salesman Problem is still the center of attention for numerous research activities. Competitions on the World Wide Web are held among students and researchers in the attempt to discover novel ways of solving the problem. The purpose of the project was to examine the feasibility of applying the emergent behavior of a Multi-Agent System to the problem.

The distributed, adaptive nature of an Multi-Agent System makes it natural to consider the possibility of extending the system, giving the user the ability to modify the graph while the system is online; letting the system dynamically adapt to the modifications made. E.g. if a user removes an edge of the graph, the system should adapt and if the edge was part of the cycle, find a new Hamiltonian cycle not including the removed edge. According to our system should be able to adapt to these modifications of the environment. This aspect of adaptivity to online modifications has to our knowledge not yet been an integral part or property of methods solving the TSP. In this light it would be very interesting to reach such a result. And not least evaluate the solutions and efficiency of the system.

REFRENCES:

[1] O. Abdoun, J. Abouchabaka, and C. Tajani, "Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem". IJES, Vol.2, No. 1, 2012.

[2] O. Abdoun, C. Tajani, and J. Abouchabaka, "Hybridizing PSM and RSM Operator for Solving NP-Complete Problems: Application to Traveling Salesman Problem". IJCSI, Vol. 9, No. 1, 2012.

[3] O. Abdoun O. and J. Abouchabaka, "A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem". IJCA, Vol. 31, No. 11, 2011.

[4] G. Reinelt, "The Traveling Salesman: Computational Solutions for TSP Applications". Berlin, Germany: Springer-Verlag, 1994.

[5] C. Walshaw, "A multilevel approach to the travelling salesman problem," Oper. Res., vol. 50, no. 5, pp. 8, 2002.

[6] G. Zaránd, F. Pázmándi, K. F. Pál, and G. T. Zimányi, "Using hysteresis for optimization," Phys. Rev. Lett., vol. 89, no. 15, p. 150 201, Oct. 2002.

[7] W. Cook, VLSI Data Sets, 2003. [Online]. Available: http://www.tsp.gatech.edu/vlsi/

[8] S. Climer and W. Zhang, "Take a walk and cluster genes: A TSP-based approach to optimal rearrangement clustering," in Proc. Int. Conf. Mach. Learn., Banff, AB, Canada, pp. 169–176, 2004.

[9] O. Johnson and J. Liu, "A traveling salesman approach for predicting protein functions," Source Code Biol. Med., vol. 1, pp. 1–7, 2006.

[10] Dorigo. M, and Gambardella. LM, "Ant colonies for the traveling salesman problem". BioSystems; 43 ; 73–81, 1997.

[11] Xiao-FengXie, "Multiagent Optimization System for Solving the Traveling Salesman Problem

(TSP)", IEEE Trans Syst Man Cybern B Cybern, VOL. 39, NO. 2, 2009

[12] Christian Agerbeck. A Multi-Agent Approach to Solving NP-Complete Problems. IMM-thesis, School of Engineering, Technical University of Denmark, 2008.

[13] Gerhard Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications, volume Lecture Notes in Computer Science, 840. Springer-Verlag, Berlin, 1994.

[14] Daniel R. Kunkle. Self-organizing computation and information systems: Ant systems and algorithms. WWW: http://www.redfish.com/dkunkle/mypapers/selfOrgAnts.pdf, 2001.

[15] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao, "An evolutionary algorithm for large traveling salesman problems," IEEE Trans. Syst.,Man, Cybern. B, Cybern., vol. 34, no. 4, pp. 1718–1729, Aug. 2004.

[16] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the traveling salesman problem. Technical Report TR/IRIDIA/1996-3, UniversitéLibre de Bruxelles, Belgium, 1996.

[17] Zar Chi Su SuHlaing, May Aye Khine, An Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem, 2011 International Proceedings of Computer Science and Information Technology, 16, 54 - 59, 2011.

[18] M. Wooldridge. An Introduction to MultiAgent Systems. Wiley, 2002.

[19] Rachid El Bejjet, HichamMedromi, A Generic Platform for a Multi-Agent Systems, Proceedings of the World Congress on Engineering and Computer Science WCECS, San Francisco, USA, 2010