2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

On the Resilience of an Ant-based System in Fuzzy Environments. An Empirical Study

Gloria Cerasela Crişan
Faculty of Sciences
Vasile Alecsandri University
Bacău, Romania
ceraselacrisan@ub.ro

Camelia-M. Pintea, Petrică C. Pop
Faculty of Sciences
Technical University Cluj-Napoca
Baia-Mare, Romania
cmpintea@yahoo.com, petrica.pop@ ubm.ro

***Abstract*—The current work describes an empirical study conducted in order to investigate the behavior of an optimization method in a fuzzy environment. MAX-MIN Ant System, an efficient implementation of a heuristic method is used for solving an optimization problem derived from the Traveling Salesman Problem (TSP). Several publicly-available symmetric TSP instances and their fuzzy variants are tested in order to extract some general features. The entry data was adapted by introducing a two-dimensional systematic degree of fuzziness, proportional with the number of nodes, the *dimension* of the instance and also with the distances between nodes, the *scale* of the instance. The results show that our proposed method can handle the data uncertainty, showing good resilience and adaptability.**

Keywords—ant system; uncertain data; fuzziness

I. INTRODUCTION

Massive data handling, communication and storage are features of the current computer processing: in 2010 the humans broke for the first time the barrier of 1 zettabyte of information and a forecast of 40 zettabytes is made for 2020 [1]. The new world - facing new data management approaches as Big Data or NoSQL, and technological developments as the Internet of Things or the cloud storage - challenges us by opening new perspectives on data processing, transfer or storage, etc.

The concept of Data Quality is extremely important nowadays, when objective characteristics (as inconsistencies, inaccuracies, lack of structure, incompatible formats) and also subjective traits (like personal norms and desires of the people interacting with such a huge amount of data) need to be assessed. One of the widely-used definitions of Data Quality describes it as the fitness for use: "The quality data meet the requirements of its authors, users, and administrators." [2]. This definition makes this concept context-dependant, so uncertain and therefore hard to formalize. Data stakeholders have separate views on what quality data means; these views may overlap, but they are never the same.

Two strategies are manifested when facing low quality data: one is to use supplementary resources (hardware [3], software, humans) in order to improve the data quality (for example the errors could be detected using hash functions [4] or corrected using cyclic redundancy check codes [5]). The other one is to use the supplementary resources in order to cope with the low quality of data – one such example is the struggle for resilient algorithms and data structures, able to tolerate some degree of errors in data without losses in correctness, performance and storage space [6, 7]. The current paper follows the second path, by investigating the stability of an optimization algorithm when fed with uncertain, fuzzy data.

The contribution of this work is two-folded: at first it describes the behavior of the chosen solving method from the stability point of view and secondly, more important, it shows that the natural features as stigmergy or synergy make the bio-inspired solving method stable and offer the premises of smooth recovery and good adaptation.

The structure of the paper follows. The second section describes the problem addressed. Section 3 presents the description of the method used to solve this problem, followed by the computation results and their analysis from Section 4. Several conclusions and research directions are shown in the last section of the paper.

II. THE PROBLEM

Most of the optimization algorithms assume that the input data are certain. This is not always the case, as faults could appear during the by-hand data input or internet file transmissions. Memory errors (bits read differently from how they were last written) could manifest due to external causes (radiations) or internal causes (permanent damage of a memory chip). Moreover, our current life needs efficient strategic (long-term) decisions, taken in extremely volatile (dynamic) environments. This means that strategies must be resilient, producing results with acceptable and predictable quality in most of the cases. Temporal Logic [8] and Semantic Web [9] are important domains that consider the same ideas.

Consequently, the results of classic optimization methods when uncertain data are fed are biased. When choosing between multiple optimization methods, one could be interested in their behavior from the stability point of view: how stable the method is, when uncertain data is used? This is the starting point of our investigation, which drove us to the following problem, central in Software Engineering:

What is the behavior of an optimization method, when the quality of the input data modifies?

In order to address this very general problem, we decided to pick a difficult optimization problem, the Traveling Salesman Problem (TSP), and to investigate the behavior of a heuristic

solving method, a variant of the Ant Colony Optimization (ACO) meta-heuristic. The reasons for these choices are multiple:

- TSP is a well-known, easy to formalize, and intensively studied NP-hard problem, taking advantage of a broad set of solving methods; there are many types of publicly-available TSP instances [10, 11]; the interest of worldwide researchers is very high, as TSP has many generalizations, including generalized TSP and generalized vehicle routing problems [12-14] with important financial and social impact.;
- the ACO class of heuristic solving methods constantly provides very good results when applied to classic TSP, manifesting fault-tolerance;
- the software package from [15] is efficient, simple to use in command line, with multiple implementations (Ant System, Elitist Ant System, MAX-MIN Ant System (MMAS), Rank-based version of Ant System, Best-Worst Ant System, and Ant Colony System [16] are available), is designed for multiple types of symmetric instances (Euclidian 2D, with explicit distances, etc.), and is offered under the GNU General Public License.

To our knowledge, the first attempt to use fuzzy data as input for an Ant-based application is [17]. Other papers on the same idea are Ref [18, 19]. Opposed to this approach, there are several works that modify the application, by introducing the fuzziness in the solving method itself [20, 21], with very promising results.

Basic descriptions of TSP and ACO meta-heuristic are given in the Section 3, which also contains the pseudocode of the method that alters the TSP instances, by introducing some degree of fuzziness in data.

Our main goal is to observe how a heuristic multi-agent application (which fundamentally offers no guarantee on the quality of the result, as exact or approximate methods do) behaves when deploying its artificial agents in an uncertain (fuzzy) environment. This behavior is compared with the situation when the same application runs in classic way (using certain data). The details of our repeated runs and the specific instances we chose from [10] and consequently transformed for our investigation needs are also presented in next section.

III. THE EMPIRICAL STUDY-METHOD

In order to illustrate the behavior of an optimization procedure when two dimensions of the data quality are cumulatively investigated, we have chosen the Traveling Salesman Problem (TSP) that basically seeks for the Hamiltonian tour with minimum length connecting all the cities on a map [22]. TSP is a difficult problem, as there are very low chances that a polynomial time solving algorithm will ever be found (it is shown that TSP belongs to the NP-Hard class of combinatorial optimization problems [23]).

The TSP is important besides its academic concern. As it has many applications in industry (drilling problems), logistics (efficient asset deployment), transportation (vehicle routing), and communications (package routing), it affects the entire society (for example, by lowering the need for transportation fuels), and receives attention from managers and decision-makers. The real life situations faced by firms rarely are perfectly and totally known – so the managers are forced to operate in imperfect knowledge, taking decisions that could have great financial and social impact. This empirical study describes the behavior of an optimization method for approaching TSP with imperfect knowledge features, so the decisional factors (human or automatic procedures) could wisely choose the appropriate solving method for the specific situation they face.

As exact methods become impractical for solving large TSP instances, we focus on a heuristic method, inspired by the way a colony of ants manages to find the shortest path from nest to food. Although the ant is an almost blind insect, which does not directly communicate with peers (instead, they deposit on the ground a specific chemical substance called pheromone, which is used as a support for indirect communication), the colony (seen like a whole) quickly and repeatedly succeeds in minimizing the travel length.

Based on the characteristics which the real ants exhibit, the Ant Colony Optimization framework describes a collection of artificial agents that cooperate and indirectly communicate in order to solve problems represented by graphs. Even most of the agents do not find very good paths, the few that do signal their successful attempt by enforcing the trace of artificial pheromone they lay when constructing their tour. The following agents most likely prefer to use the edges with high amount of pheromone, and so they more likely use the shorter paths, exploiting the previous knowledge [16].

TSP was the first problem tackled with ACO methods. Today, ACO metaheuristic is successfully applied to static and dynamic problems in scheduling, transportation, assignment, routing, folding, classification, data mining, communications, image processing etc. [24-28].

With such a broad range of interest raised by TSP, the researchers designed and coded several successful TSP solvers. Concorde [29] is the most-known exact solver, used for computing the optima for all the instances from [10] (free for academia use). LKH solver [30] is a state-of-the-art implementation of the heuristic described in [31]. Google Maps API developers could base their applications for route planning on (under a MIT license) the code from [32]. The current paper uses the ACOTSP implementation [15] created and maintained by the author of MAX-MIN Ant System, a very successful ACO method [33].

Fuzzy numbers generalize the numbers, expressing the possibility (the vagueness), as random values express the probability. The fuzzy number is a set of possible values, each with its own weight between 0 and 1 [34]. Like classic numbers, fuzzy numbers support algebraic operators, and can easily be handled in applications as data structures. If the result of an application designed for fuzzy numbers need to be compared with the result offered by an application handling numbers, then a defuzzification final step (providing a particular number) is needed for the first application. Conversely, as is used here, one could choose to model the

fuzzy data by repeating the executions of a classic application and different values of entry data, and taking a statistic of the set of results.

The fuzziness is introduced here by modeling the real life situations, when local events (like bad weather, road closing, broken car) could affect the travel cost (measured in time, length, or money). Instead of using *a single value* for each edge length, we executed the same application several times, with entry data slightly modified: some selected nodes glided in their neighborhood, in order to explore the impact of *other possible values* for the distances connecting those nodes with all the other nodes.

As entry data 32 Euclidean 2D instances from [10] are used, with number of nodes between 51 and 439. The basic entries were modified in order to introduce some degree of fuzziness, *affecting each instance in the same way*. After several attempts we have chosen an altering function that takes as input a TSP instance *inst* from [10] and two parameters denoted by *a* and *b* and returns a modified instance *new-inst* as follows: *new-inst* is an Euclidean 2D instance, with the same number of nodes as *inst*, but some nodes having different positions on the map.

Two parameters were used for *globally* expressing the fuzziness of an instance (both *a* and *b* store positive integer values, less than 100):

- the *dimension* regularity: how many nodes change their position with the same impact irrespective of the number of nodes; randomly *a*% of nodes move;
- the *scale* regularity: how far these nodes are moved for maintaining the same impact irrespective of the distance from one selected node to the nearest; each node *i* is randomly relocated in a disc with the center in the old position of *i* and the radius the product between *b*% and the smallest distance from *i* to other possible nodes. We used the same idea as in [35].

The pseudocode for the function *Alter* introducing the fuzziness in data is presented next.

```
Function Alter (instance inst, int a, int b)
  n = nodes(inst)
  k = n * a/100
  repeat k times
     randomly choose an unselected node i
     from inst
     x is the minimum of the distances from i
     to any other nodes from inst
     y = x * b/100
     randomly choose a new position for i
     within the disc with center i, radius y
  end-repeat
  new-inst = inst with new positions for all k
  nodes previously chosen
return new-inst
```

The design of our experiment is presented in Figure 1. It starts with *inst*, one TSP classic instance. The procedure *Alter* already described transforms it in *new-inst*, a new "close" instance, depending on *inst*, *a* and *b*. The original instance and the new, modified ones are independently fed into the classic MMAS, and the results are statistically analyzed.

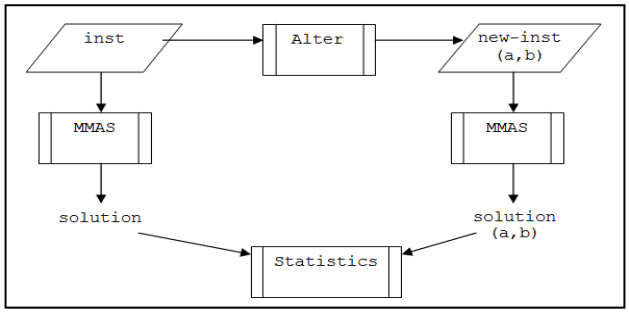

Fig. 1. Experiment design

IV. COMPUTATIONAL RESULTS

As specified in the previous Section, our empirical study uses the ACOTSP package from [15], executed both on the basic instances from [10] and on the instances returned by the procedure *Alter*, with $a \in \{10,25\}$ and $b \in \{25,50\}$. The results of these tests are further illustrated in Table 1. For each TSP instances of first column, five runs are made using the original dataset [10] and four modified datasets, using a *close-distance* move (b = 25%) or a *medium-distance* move (b = 50%) performed on *few* nodes (a = 10%) or on a *medium* number of nodes (a = 25%).

Each time the MAX-MIN Ant System [33] used ten artificial ants, with a time limit of sixty seconds for any iteration. The distinct features of MMAS are:

- only the ant that found the (historical) best tour is allowed to deposit pheromone;
- the amount of pheromone on each edge is always lower- and upper-bounded;
- at the beginning, all the edges receive the maximum pheromone quantity [33].

The MMAS implementation [15] also uses *3-opt*, a powerful local search procedure, that repeatedly tries to lower the length of a tour by deleting 3 edges and replacing them (if this lowers the total length) with the best re-connection from all the possible ones [31].

As we intended to collectively address the behavior of the considered Euclidean-2D TSP instances [10] the following data were collected: the best tour found in ten iterations, the average of the best tours, the average step that finds the best value each time, the standard deviation of the best values found and the standard deviation of the iterations that find the best values, taken from ten iterations. Based on these data, the standard deviation between the best value found for each specific pair of values of parameters *a* and *b*, and the optimum known for the classic variant of each instance were computed. The last four columns of Table 1 describe the deviation from optimum when a data uncertainty is introduced – one column for each possible combination when $a \in \{10,25\}$ and $b \in \{25,50\}$.

As a global behavior, the average value for each column was computed on the last line. The small difference between the average values for $a = 10$ (columns 2 and 3 - Table 1) show the stability of the application: few perturbations have almost the same impact, even if their scale differs. As the average value from column 3 is less than the corresponding one from column 4, we can say that is more desirable from the stability point of view to have few large data uncertainties, instead of having more small ones. The very large last value from Table 1 entitles us to conjecture that the two dimensions of inconsistency do not linearly compose. The effect of combining both high levels of inconsistencies is dramatic: the average deviation more than doubles when only *a* changes and *b* is high, and almost doubles when only *b* changes and *a* is high. This means that when a certain threshold is exceeded, the system begins to express unstable behavior.

TABLE 1 STANDARD DEVIATION BETWEEN BEST VALUES AND OPTIMAL

Instances	StdDev(%) (a, b)			
	(10, 25)	*(10, 50)*	*(25,25)*	*(25, 50)*
ch130	0.0283	0.0849	0.2758	0.8061
ch150	0.1344	0.6223	0.0849	0.5303
eil51	0.0424	0.1273	0.0141	0.0566
eil76	0.0212	0.0849	0.0071	0.0636
eil101	0.0071	0.0283	0.0495	0.0424
gil262	0.0778	0.1344	0.0424	0.0495
kroA100	0.5091	2.7506	0.7212	0.8627
kroB100	0.2121	0.2687	0.0778	1.9516
kroC100	0.6859	1.0819	0.6081	0.6152
kroD100	0.2899	0.2758	0.9829	1.9870
kroE100	0.5445	0.0849	1.3294	0.5515
kroA150	1.1809	0.1768	0.8768	0.2616
kroB150	0.0849	0.2828	0.4313	0.8273
kroA200	0.3182	0.6647	5.5720	2.5032
kroB200	0.0000	0.0000	0.8061	1.2587
lin105	0.4101	0.3818	0.1626	0.8202
lin318	1.0182	0.6576	1.9304	0.7071
pr76	1.4708	1.5768	1.4779	3.9739
pr107	0.3465	0.6010	0.7566	0.3677
pr124	0.2616	1.5274	0.5020	2.7931
pr136	1.9304	4.3275	2.4607	16.2635
pr144	0.6010	0.4596	0.8485	0.5303
pr152	2.2132	0.1980	1.7041	1.0889
pr226	2.0930	0.8273	0.2828	4.8154
pr264	0.8485	0.7142	0.9263	0.0990
pr299	0.4738	0.5869	0.3960	1.3364
pr439	2.4112	1.8243	0.0000	6.8165
rd100	0.1344	0.2687	0.3111	0.1061
rd400	0.3536	0.2616	0.4172	0.5162
st70	0.0212	0.0071	0.0424	0.0283
ts225	1.3435	2.5244	5.6922	3.0406
tsp225	0.0424	0.0141	0.1061	0.1344
Average	***0.6284***	***0.7321***	***0.9343***	***1.7439***

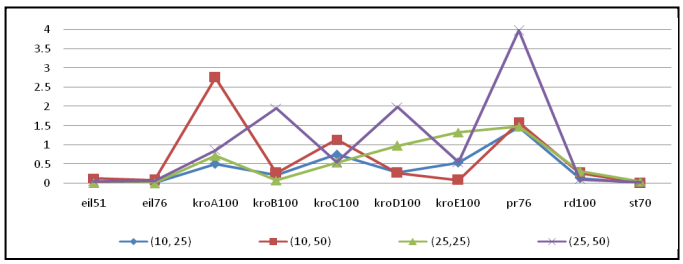

Fig. 2. Standard deviations: instances with dimensions between 50 and 100

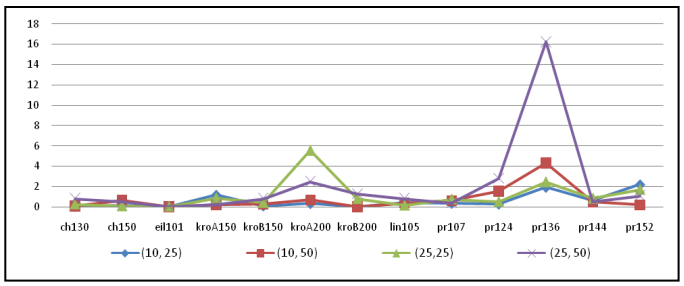

Fig. 3. Standard deviations: instances with dimensions between 100 and 200.

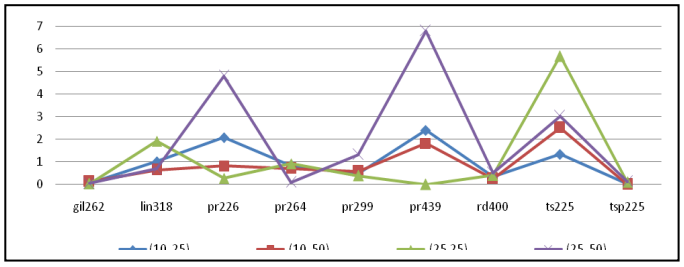

Fig. 4. Standard deviations: instances with dimensions greater than 200.

At a global level too, we notice that 115 from the total number of the transformed instances ($32 \cdot 4 = 128$) were solved each time to the optimum, showing the method stability as it each time (from 10 iterations) finds the optimum in 90% of the cases.

Figures 2-4 illustrate the results based on the instance dimensions. For Figure 2, with the number of nodes between 51 and 100, the uncertainty can be considered at a low level and persistent. Figure 3 presents the instances with the number of nodes between 101 and 200, and exhibits a low uncertainty, clustered for the group *kro* and *pr*; the instance *pr136* has an extreme result, and it was excluded from the group. The large instances with more than 200 nodes, shown in Figure 4 have a systematic higher uncertainty, mainly the largest instance *pr439* and *ts225*, specially designed to be difficult.The next step of the investigation was to consider the variation on one dimension of uncertainty: the standard deviations between the results when one parameter has a constant value, and the optimum value for the classic variant were computed (Table 2). For example, the column 2 holds the standard deviation between the results when $a = 10$ and $b \in \{25,50\}$ and the optimum.

The last line represents again the average values for each column. Its values show that, at a global level, the sensitivity along the first dimension of the uncertainty (how many nodes do not have certain data) is higher than the sensitivity along the second dimension (how big the uncertainty is). This conclusion came up as the interval between the first two values (meaning that *a* can vary) is bigger than the interval between the last two ones (when only *b* can vary). So, if the programmer can lower the data uncertainty, then it is better to spend computing resources on totally eliminating some uncertainties, instead of lowering their average scale (of course, at an affordable trade-off). But, when uncertainty becomes low, it is better to concentrate on its scale, as the last

value from column 4 is higher than that from column 2. Again, there is a threshold that can orient the programmer's effort, and so the application can perform better with the same computing costs.

TABLE 2 STANDARD DEVIATION WITH A CONSTANT PARAMETER.

Instances	StdDev(%) (a, b)			
	(10, 25)	*(10, 50)*	*(25,25)*	*(25, 50)*
ch130	0.0611	0.5794	0.2376	0.6954
ch150	0.4631	0.4029	0.1563	0.8159
eil51	0.1249	0.0416	0.0416	0.0902
eil76	0.0624	0.0493	0.0153	0.0624
eil101	0.0265	0.0379	0.0379	0.0503
gil262	0.0954	0.0651	0.0551	0.1345
kroA100	2.0696	0.6543	0.5242	1.9895
kroB100	0.2003	1.6262	0.2122	1.4959
kroC100	0.7741	0.4994	0.9156	1.2150
kroD100	0.4000	1.4050	0.7142	1.7459
kroE100	0.4143	1.3674	1.3632	0.4200
kroA150	0.9007	0.8433	0.8671	0.3119
kroB150	0.2053	0.9046	0.3232	0.8159
kroA200	0.7092	3.9468	4.6848	1.8337
kroB200	0.0000	0.9016	0.6582	1.0277
lin105	0.3239	0.6142	0.2921	0.5805
lin318	0.7301	1.3812	1.3657	0.5582
pr76	1.2464	3.9874	2.0850	2.8299
pr107	0.4267	0.8107	0.5356	0.6916
pr124	1.3665	2.5107	0.3551	1.9779
pr136	3.0659	12.3972	1.8315	11.9121
pr144	0.7522	0.9836	1.0300	0.7006
pr152	1.7319	1.2204	2.7778	0.9800
pr226	1.4908	3.8215	1.6060	3.6413
pr264	0.6450	0.7998	0.7267	0.6275
pr299	0.7514	1.2838	0.3593	0.9473
pr439	3.0046	5.5657	1.9688	6.4409
rd100	0.2902	0.2237	0.2207	0.2732
rd400	0.4366	0.3874	0.5456	0.3650
st70	0.0208	0.0306	0.0300	0.0265
ts225	1.7862	4.0281	4.2078	2.3010
tsp225	0.0306	0.1002	0.1082	0.1159
Average	***0.7690***	***1.6710***	***0.9641***	***1.4898***

V. CONCLUSIONS AND FUTURE WORK

The main goal of our work is to measure *the bias* (the average distance from the optimum in the classic way) and *the variance* (the variation of these distances when the application runs for ten times) when two types of uncertainty are considered for several Euclidean TSP instances, with number of nodes ranging from 51 to 439 and with different scales. Two parameters for implementing the data uncertainty were used to globally study such diverse instances.

The computational results show a general resilience and adaptability of the ant-based solving method. We also observed that is more desirable to have few large uncertainties, instead of having many small uncertainties, and that the two dimension of uncertainty do not linearly compose.

At each dimension of uncertainty level, our experiment shows that it is desirable to totally eliminate some uncertainties, instead of lowering their average scale.

When the resilience of an application tackling uncertain data is apriori known, the computing resources can be intelligently used. The paper describes such a behavior and the decision-makers can use it when choosing between several optimization methods.

As future investigation, we plan to add the dimension of time to data uncertainty and to study the behavior of ant-based application when the data fuzziness goes dynamic. Also further investigations on other meta-heuristics including bio-inspired computing techniques will be developed.

ACKNOWLEDGMENT

Author G.C. Crişan. This paper was developed within the project "Bacau and Lugano - Teaching Informatics for a Sustainable Society", co-financed by Switzerland through the Swiss-Romanian Cooperation Programme to reduce economic and social disparities within the enlarged European Union.

Author P.C. Pop. The work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-RU-TE-2011-3-0113.

REFERENCES

[1] J. Gantz, D. Reinsel, "*The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East*", Framingham: IDC iView, Analyze the Future, 2012.

[2] M. J. Eppler, *Managing Information Quality: Increasing the Value of Information in Knowledge-intensive Products and Processes*, 2nd Ed, New York/Heidelberg: Springer, 2006.

[3] M. M. Balas, V.E. Balas, *World Knowledge for Control Applications*, 11th IEEE International Conference on Intelligent Engineering Systems, 225--228, 2007

[4] T. H. Cormen, C.E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (3rd ed.), Cambridge, MS: MIT Press, 2009.

[5] W. W. Peterson, E. J. Weldon, *Error-correcting codes*. 2nd Ed. Cambridge, MS: MIT Press, 1972.

[6] J. von Neumann, *Probabilistic logics and the synthesis of reliable organisms from unreliable components,* in C. Shannon, J. McCarty (Eds.), *Automata Studies*, Princeton University Press, 1956, pp. 43–98.

[7] I. Finocchi, F. Grandoni, G. F. Italiano, "Designing Reliable Algorithms in Unreliable Memories Algorithms", *Lecture Notes in Computer Science* vol. 3669, 2005, pp 1-8

[8] P. Øhrstrøm, P. F. V. Hasle, *Temporal logic: from ancient ideas to artificial intelligence*. New York/Heidelberg: Springer 1995.

[9] *Semantic web*. Available: http://www.w3.org/standards/semanticweb/

[10] *Library of sample instances for the TSP.* Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[11] *8th DIMACS Implementation challenge: The Traveler Salesman Problem*. Available: http://dimacs.rutgers.edu/Challenges/TSP/

[12] C-M. Pintea, P. Pop, C. Chira, "The Generalized Traveling Salesman Problem solved with Ant Algorithms", Computing Research Repository 1310.2350, 2013.

[13] C-M. Pintea, C. Chira, D. Dumitrescu, P.C. Pop, "Sensitive Ants in Solving the Generalized Vehicle Routing Problem", *Int. J. Comput. Commun.*, vol. 6, no. 4, pp.731-738, 2011.

[14] C-M. Pintea, P.C. Pop, D. Dumitrescu, "*An Ant-based Technique for the Dynamic Generalized Traveling Salesman Problem",* 7-th Int. Conf. on Systems Theory and Scientific Computation, pp.257-261, 2007.

[15] *ACO public software*. Available: http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.html

[16] M. Dorigo, T. Stüzle, *Ant Colony Optimization*, Cambridge, MA: MIT Press, 2004.

[17] G. C. Crişan, E. Nechita, "Solving fuzzy TSP with Ant algorithms", *Int. J. Comput Commun*, vol. 3(S), no. 3, pp. 228-231, 2008.

[18] A. A. Alsawy, H. A. Hefny, F. El-licy, "Fuzzy ant colony optimization algorithm". *Informatics and Systems (INFOS),* pp. 1-5, 2010.

[19] S.A. Khan, A.P. Engelbrecht, "A fuzzy ant colony optimization algorithm for topology design of distributed local area networks," *IEEE Swarm Intelligence Symposium, 2008.* pp. 1-7, 2008.

[20] A. R. G, Ginidi, A. M. A. M. Kamel, H. T. Dorrah, „Development of new Fuzzy Logic-based Ant Colony Optimization algorithm for combinatorial problems", in *Proc. of the 14th International Middle East Power Systems Conference,* Cairo, Egypt,. 2010.

[21] A. George, B. R. Rajakumar, "Fuzzy aided Ant Colony Optimization algorithm to solve optimization problem". *Advances in Intelligent Systems and Computing* vol. 182, pp. 207-215, 2013.
[22] W. J. Cook, *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press, 2012.
[23] R. M. Karp, "Reducibility among Combinatorial problems". in R.E. Miller, J.W. Thatcher (Eds.), *Complexity of Computer Computations*. The IBM Research Symposia, pp. 85-103, NY: Plenum. Press 1972.
[24] C. M. Pintea, *Advances in Bio-inspired Computing for Combinatorial Optimization Problem*, Springer 2014.
[25] P. C. Pop, *Generalized Network Design Problems. Modeling and Optimization*. Walter de Gruyter, 2012.
[26] E. Nechita, C. Muraru, M. Talmaciu, "Mechanisms in social insect societies and their use in Optimization. A case study for trail laying behavior", in *Proc. of the 1st International Conference Bio-Inspired Computational Methods Used for Solving Difficult Problems*, Tg.Mureş, Romania, 2008, AIP Proceedings, NY, 2009, pp. 171-179.
[27] P. C. Pop, C. M. Pintea, C. P. Sitar, "*An Ant-based Heuristic for the Railway Traveling Salesman Problem"*, Lecture Notes in Computer Science vol. 4448, pp. 702-711, 2007.
[28] G. C. Crişan, C. M. Pintea, C. Chira, "Risk assessment for incoherent data", *Environ. Eng. Manag. J.*, vol. 11, no. 12, pp. 2169-2174, 2012.
[29] *Concorde TSP solver*. http://www.math.uwaterloo.ca/tsp/concorde/
[30] *LKH TSP solver*. http://www.akira.ruc.dk/~keld/research/LKH
[31] S. Lin, "Computer solutions of the Traveler Salesman Problem", *Bell System Tech J.* vol. 44, no. 10, pp. 2245-2269, 1965.
[32] *Google TSP solver*. https://code.google.com/p/google-maps-tsp-solver/
[33] T. Stützle, H. Hoos, "MAX-MIN Ant System", *Future Generation Comp. Sys.* vol. 16, no. 8, pp. 889-914, 2000.
[34] L. A. Zadeh, "Calculus of fuzzy restrictions", in L. Zadeh, K. Fu, K. Tanaka M. Shimura (Eds.), *Fuzzy Sets and their Applications to Cognitive and Decision Processes*, NY:Academic Press, 1975, pp.1-39.
[35] A. L. Barabási, E. Bonabeau, "Scale-Free Networks", *Scientific Amer.,* vol. 288, no. 5, pp. 60–69, 2003.