Adaptability of a Discrete PSO Algorithm applied to the Traveling Salesman Problem with Fuzzy Data

Camelia–M. Pintea
Faculty of Sciences
Technical University Cluj-Napoca
Baia-Mare, Romania
dr.camelia.pintea@ieee.org

Simone A. Ludwig
Department of Computer Science
North Dakota State University
Fargo, ND, USA
simone.ludwig@ndsu.edu

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

***Abstract*—Imperfection is a common characteristic of information nowadays. For example, in everyday life, decisions have to be made based on information that is incomplete, inconsistent, and/or uncertain. This inexactness makes the decision making a challenging task. This paper investigates the behavior of a well-known optimization method, *Particle Swarm Optimization (PSO)*, when solving a fuzzy problem. The discrete *PSO* implementation is studied on a *Traveling Salesman Problem (TSP)* variant, designed to model the uncertain environmental influences. The experiments investigate several symmetric TSP instances and their fuzzy variants in order to study the impact of uncertain information in the quality of the results provided by *PSO*. The fuzzy variants were generated using a two-dimensional degree of fuzziness, which is proportional to the number of nodes of the instance. In addition, the amplitude of the uncertainty can be set at running time, so the degree of fuzziness used here is a systematic perturbation, providing similar effects on all studied *TSP* instances. The experimental results reveal that the *PSO* algorithm can handle uncertainty in data by showing good adaptability based on the used *TSP* benchmark set.**

Keywords—Traveling Salesman Problem; Particle Swarm Optimization; Uncertainty Model.

I. INTRODUCTION

One of the common characteristics of information available to humans is its imperfection. Information can be incomplete, inconsistent, uncertain, or all three combined. The process of decision-making becomes a very difficult task since the information for solving a problem is often inexact. However, a human expert can usually cope with these imperfections, can make correct judgments, and can make the right decisions even in complex situations.

Uncertainty is defined as the lack of exact knowledge that would enable one to reach a perfectly reliable conclusion [1]. Classical logic, which permits only exact reasoning, where perfect knowledge always exists, and the law of the excluded middle always applies, do not consider reasoning under incertitude [2]. Unfortunately, most real world problems do not embed clear-cut knowledge. The available information often contains inexact, incomplete or even immeasurable data. This is why approaching such real-world situations must rely on new solving paradigms, able to efficiently operate – like a human mind does – with uncertain data.

In general, we can identify four main sources of uncertain knowledge: weak implications, imprecise language, unknown data, and the difficulty of combining the views of different experts [3].

Uncertainty in data collected via experiments, sensor data, the *World Wide Web,* etc., raise serious challenges to decision making and reasoning, in particular when the data volumes are high, and the problem restrictions are difficult to evaluate. Uncertainty in data leads to wrong outcomes, decisions, and judgments. Therefore, meeting data quality standards is essential; standardization, validation, and enhancement are several methods to succeed in improving the data quality. The processes of standardization and validation clean data, whereas enhancement is the result of the successful application of standardization and validation. Enhancement gives value and usability to the data [4].

Researchers when facing data of low quality have proposed several strategies. One strategy is to use extra hardware, software, or human resources for improving the data quality [5]. This goal can be reached by hash functions [6] to detect errors, cyclic redundancy check codes [7] for correcting errors, etc. Another direction is to use the same resources in order to design more complex and reliable applications that can handle uncertain events. One such example is the category of resilient algorithms: they are able to tolerate some degree of errors or failures in data without the loss of correctness, performance, and storage capability [8][9]. Different approaches in processing uncertain databases are presented in [10].

This paper investigates the effect of an evolutionary computation method when the optimization task contains fuzzy data. In particular, the resilience of a biologically-inspired algorithm is studied when low quality data are involved. This work focuses on the *Particle Swarm Optimization (PSO)* approach, and investigates its sensitivity to fuzzy data, continuing the practical experiments from [11]. A comparison from the adaptability point of view between *PSO* and the previously proposed *Ant Colony Optimization* approach is also performed.

The paper is organized as follows. Related work is outlined in Section II. In Section III, the *PSO* approaches as well as the fuzzification method for the Traveling Salesman Problem (*TSP*) are described. Section IV contains the experimental results obtained, and Section V concludes the paper with the outcomes of this study.

II. Related Work

The *Traveling Salesman Problem (TSP)* [12] is the optimization benchmark problem used to investigate the impact of fuzzy data applied to an evolutionary computation method referred to as the *Particle Swarm Optimization (PSO)* technique.

The *Traveling Salesman Problem* is described as follows:

Given a complete graph with weights on the edges (arcs), find a Hamiltonian cycle with minimum total weight.

The usual representation for *TSP* is a 2D map, where the nodes are the cities, the edges are the roads connecting the cities, and the weight for an edge is the distance between the corresponding cities. The goal for the salesman is to find one shortest path for visiting all the cities and coming back to the starting city.

The *TSP* is known to be a complex and difficult problem [13]. With broad applications in transportation, logistics, industry, communications, etc., it is under heavy investigation by the research community. *TSP* has multiple variants, designed for better reflecting real situations: asymmetric *TSP (aTSP*) when the distance function is not symmetric, *multiTSP (mTSP*) when multiple salesmen are available for completing partial tours that cover all the cities from the map, etc.

Many publicly available instances of the *TSP* are available [14][15]; as well as many real-life complex problems and instances are given in [16][17]. All of those assume that the problem is exact and all data are integrally known at the solving time.

Uncertain *TSP* can be modeled in several ways. The first attempt to introduce uncertainty was presented in [18]. The *Probabilistic TSP* seeks a most efficient *a-priori* tour when each city is to be visited with a specific probability. Fuzzy *TSP* is defined as *TSP* with fuzzy numbers as distances on the edges [19]. The tours in this case are classic tours with fuzzy numbers as lengths. This *TSP* variant needs supplementary decisions, since fuzzy numbers are not totally ordered as real numbers are. The *TSP with interval data* specifies the distance between any two cities as the range of possible values, and no additional assumption is made [20]. The solution to this *TSP* variant uses the concept of *robust deviation*: the best tour minimizes the maximum deviations over all realizations of edge distances. *Dynamic TSP* models the changing situations in the world: the distance function is dynamic, and so is the number of cities [21].

Of course, combinations of new features are constantly designed and investigated. For example, a dynamic, clusterized *TSP* version is investigated in [22]. An uncertain multiobjective *TSP* version is defined and solved using Genetic Algorithms in [23].

Many difficult problems can be seen as *TSP* generalizations. One such example is the *Vehicle Routing Problem* (VRP) [24]. The current problems are so complex and rich in attributes that researchers investigate new methodologies for efficiently solving them when computing resources are scarce [25].

During the past decades several successful algorithms were proposed to solve the TSP problem. For example, Concorde [26] is an exact solver used to compute the optimal value to 106 from the 110 *TSPLIB* instances in [14]. Efficient parallel branch-and-bound methods are reported for example in [27]. Other successful techniques for TSP are the *Ant Colony Optimization* (*ACO*) [28] or the *Particle Swarm Optimization* (*PSO*) [29].

Ant Colony Optimization (ACO) is a biologically-inspired heuristic solving method for optimization problems represented by a weighted graph. It uses a population of artificial agents that seek to construct a least cost tour, modeling the way a colony of ants manage to find the shortest path from the food to the nest. The balance between space exploration and the exploration of the good paths already found by concurrently acting artificial ants, together with the broad set of problems that allow *ACO* implementations, made this approach intensively used.

ACO and *PSO* belong to the *Swarm Intelligence* domain, which deals with natural and artificial systems composed of many coordinated individuals that complete a difficult common task using decentralized control and self-organization. The expression was introduced in [30], in the context of cellular robotic systems. *PSO* is presented in the following section.

III. Proposed Method

The effect of a discrete *PSO* algorithm is investigated on the *TSP* benchmark with perfect and uncertain/fuzzy data. The presentation of the discrete *PSO* algorithm is followed by the description of the fuzzified *TSP* benchmark instances.

A. Discrete PSO Algorithm

Particle Swarm Optimization (*PSO*) was initially defined for continuous problems. The idea behind *PSO* is to model the way a bird flock globally moves [29]. It was developed for a broad range of problems with large solution spaces; it uses simple mathematical relations based on particle position and speed, and it delivers very good results for a broad range of difficult problems [31]. The discretization of *PSO* can be done in multiple ways, depending on the problem structure and type. For example, a discrete implementation of *TSP* is described in [32]. The discrete *PSO* equations are as follows:

$$x(t+1) = x(t) + v(t+1) \tag{1}$$

$$v_{ij}(t+1) = c_1 v(t) \oplus c_2\left(x_{Best}(t) - x(t)\right) \oplus c_3(t)\left(x_{GBest}(t) - x(t)\right) \tag{2}$$

where x represents a particle, i denotes the particle's number, j the dimension, t a point in time, and v is the particle's velocity. The variable x_{Best} is the best location the particle ever visited (the particle's knowledge), and x_{GBest} is the best location any particle in the swarm ever visited (the swarm's knowledge). The value of c_1 represents the inertia weight and used to weigh the previous velocity, c_2 is a variable to weigh the particle's knowledge, and c_2 is a variable to weigh the swarm's knowledge.

More information on the *PSO* implementation for *TSP* regarding the multiplication, addition, and subtraction

operators are found in [32]. Details on a *PSO* approach to a *VRP* with uncertain demands are presented in [33]. Multi-objective *PSO* can be used for designing efficient robot paths in uncertain environments [34]. A multi-swarm approach for solving dynamic optimization problems is presented in [35]. The complexity of dynamic problems is tackled in [36] by a two-scale framework with low-level *PSO* classic operators, and high-level new operators, able to enhance the particle diversity and to avoid local convergence.

B. Traveling Salesman Problem Benchmark

The *TSP* is used as the benchmark to investigate the behavior of the *PSO* algorithm on an uncertain problem. In particular, the fuzziness impact as an imperfect knowledge characteristic is studied. Fuzzy logic is employed in order to generate the uncertainty for the benchmark. Fuzzy numbers are used to represent the vagueness, the same way that random values express the probability.

The new mathematical theory of fuzzy sets was developed by Lotfi A. Zadeh in [37]. Zadeh introduced the *fuzzy sets*, where many degrees of membership are allowed, and indicated with a number in [0,1]. The point of departure for fuzzy sets is the generalization of the valuation set from the pair of numbers {0,1} to all the numbers in [0,1]. This is called a *membership function*, denoted as $\mu_A(x)$ and in this way defines the fuzzy set A. An alpha cut (α-cut) is a crisp set of elements of A belonging to the fuzzy set to a degree at least α. Further details and representation theorems for fuzzy concepts are given in [38].

This fuzzification process is applied to the *TSP* benchmark that models real situations when the travel cost of the salesman on several roads is affected by external events such as sudden weather changes, or transportation difficulties like a car engine failure.

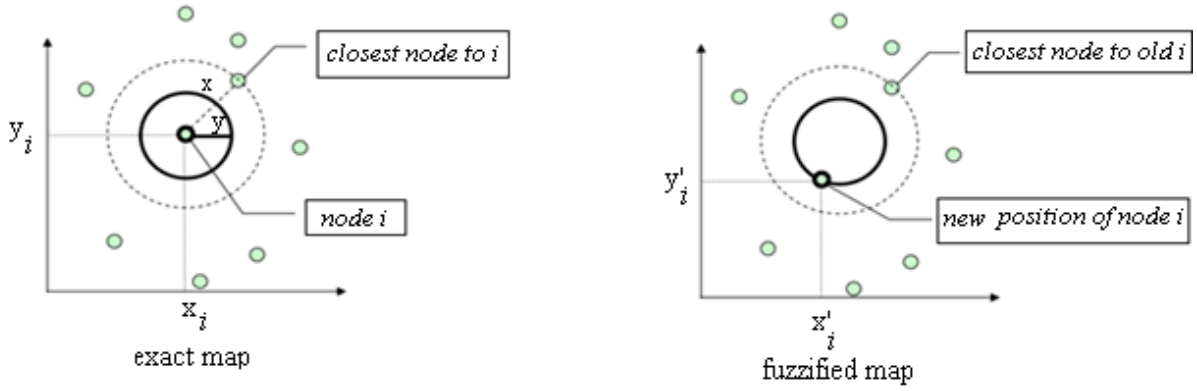

Figure 1. Representation of an initial map (left side) and its fuzzified form (right side)

The entry data of the fuzzy *TSP* variant we study here consist of slight and systematic modifications of the position of several nodes within their neighborhood. Another approach to fuzzy *TSP* is presented in [39].

The initial benchmark data is fuzzified by applying the same degree of fuzziness to each instance. The *TSP* instance *inst* taken from [14] is used as input and the modified instance has the same number of nodes, but some nodes are positioned differently on the map. Figure 1 shows an example of obtaining a fuzzified *TSP* instance (right side) from an exact initial instance (left side).

Two global integer parameters $a \in (0, 100)$ and $b \in (0, 100)$ are used to express the uncertainty, i.e., the fuzziness for each instance. The parameter a is called the dimension regularity, specifying how many node changes are applied (with the same impact irrespective of the number of nodes). The parameter b is called the scale regularity, which specifies how far the nodes move (maintaining the same impact irrespective of the distance). Thus, each node is relocated in a random order within a circle as shown in Figure 1. The figure shows the modified position of node i, randomly chosen using $C(i, y)$, the circle with the center in the current node i and the radius $y=x*b/100$, where x is the distance from i to the nearest node.

```
function fuzzifyInstance(inst, a, b)
n = count_nodes(inst)
k = n*a/100
for j=1 to k do
   Randomly choose an unvisited node i∈inst
   x = min{d(i,u), u∈inst, u≠i}
   y = x*b/100
   Randomly choose a new position i'∈C(i,y)
end for
return inst
```

Algorithm 1. The fuzzification function `fuzzifyInstance`

Algorithm 1 shows the pseudo code of the *fuzzifyInstance* function. The function returns the modified instance *inst* after applying the modifications controlled by the values of the parameters a and b. The function *min* returns the minimum of the distances between the current node i and any other node from *inst*, and $C(i, y)$ is the circle with the center being the current node i and radius y.

IV. EXPERIMENTAL RESULTS

To test our fuzzification method, several instances from the TSPLIB [14] library are used: *Krolak/Felts/Nelson* with 100 nodes, *Christofides/Eilon* with node dimension between 51 and 101, and *Padberg/Rinaldi* instances with node dimension between 76 and 124. The computer used had the following processor specifications: AMD with 2.8GHz and 3GB of RAM.

These instances were modified based on the input parameters *a* and *b* specified in the description of the *fuzzifyInstance* function. From the 11 initial instances we derived 44 fuzzified instances; for each original instance four new, fuzzified instances resulted, corresponding to the four following sets of new parameters *a* and *b*: {a=10%, b=25%}, {a=10%, b=50%}, {a=25%, b=25%} and {a=25%,b=50%}.

The original instances and their fuzzified variants were solved using the exact method with CPLEX [25], *ACOTSP* [40], and *PSO* [29]. The applications were executed with implicit values for the running parameters. As many results show [41][42], the parameter settings are very important for heuristic methods, and we preferred to use their implicit values, recommended by the *ACO* and *PSO* designers. For the *ACO* implementation, we used the *MAX-MIN Ant System* (*MMAS*) with no local search, as *PSO* has not such a supplementary method. The results are presented in Table 1 (for the executions with the new instances derived when a=10%), and in Table 2 for the two sets of instances obtained when a=25%.

For each fuzzified instance we measured the impact of the data fuzzification introduced by the function *fuzzifyInstance* through the new parameter PE (percentage error):

$$PE = \frac{solution - best}{best} \cdot 100\% \qquad (3)$$

where *solution* is the best solution (in ten runs) found by the two heuristic methods on the fuzzified instance, and *best* is the optimal solution.

Table 1. Comparison of *PE* values in % for *CPLEX*/*ACO*/*PSO* method, for *a*=10, *b*∈{25,50}.

(a, b) Application / Instances	(10, 25)			(10, 50)		
	CPLEX	ACO	PSO	CPLEX	ACO	PSO
Krolak/Felts/Nelson instances						
Average	-0.03	2.40	-0.08	-0.30	3.81	-2.54
Minimal	-0.35	-1.83	-3.58	-1.83	1.45	-9.69
Maximal	0.47	6.25	1.29	0.74	5.99	4.23
Christofides/Eilon instances						
Average	0.23	0.34	0.43	-1.94	-1.70	-3.09
Minimal	-0.56	-0.56	-0.33	-4.23	-3.99	-7.98
Maximal	1.41	1.41	1.20	0.64	1.11	-0.13
Padberg/Rinaldi instances						
Average	-0.08	0.24	-0.44	-0.25	0.46	-1.72
Minimal	-0.19	-0.15	-2.22	-0.37	0.07	-5.45
Maximal	0.06	0.48	0.60	-0.19	0.82	0.51
Overall Average	***0.03***	***1.25***	***-0.03***	***-0.73***	***1.39***	***-2.45***

Table 2. Comparison of *PE* values in % for *CPLEX*/*ACO*/*PSO* method, for *a*=25, *b*∈{25,50}.

(a, b) Application / Instances	(25, 25)			(25, 50)		
	CPLEX	ACO	PSO	CPLEX	ACO	PSO
Krolak/Felts/Nelson instances						
Average	0.60	5.63	1.10	-0.10	3.98	0.13
Minimal	-0.64	1.92	-1.10	-1.22	2.26	-2.80
Maximal	2.38	8.04	3.63	0.94	5.73	4.25
Christofides/Eilon instances						
Average	-0.26	3.98	-0.05	-1.50	2.82	-5.94
Minimal	-0.79	1.88	-4.79	-2.58	-1.17	-10.02
Maximal	0.00	7.63	4.29	-0.79	9.06	-0.27
Padberg/Rinaldi instances						
Average	0.02	5.38	-0.23	-0.04	3.80	-1.06
Minimal	-0.16	1.56	-1.31	-0.89	0.71	-4.16
Maximal	0.18	7.91	0.61	0.90	6.14	0.91
Overall Average	***0.21***	***5.11***	***0.27***	***-0.47***	***3.61***	***-2.29***

The *TSP* instances used are from the groups: *Krolak/Felts/Nelson* group: *kroA100, kroB100*, *kroC100, kroD100* and *kroE100*; the *Christofides/Eilon* group: *eil51, eil76*, and *eil101*, and the *Padberg/Rinaldi* group gathers *pr76, pr107*, and *pr124*.

For each group, each application, and each set of the new parameters *(a, b)* we report in Tables 1 and 2 three numerical values based on the *PE* computed by formula (3): the average, the minimum, and the maximum for the *PE* values. The last lines from the Tables 1 and 2 hold the average values of the variable *PE* taken on all the eleven fuzzified instances.

Figure 2 represents the numerical values taken from the last lines from Tables 1 and 2: the average influence of the data fuzziness for each set of the new parameters *a* and *b*. The variations in the results show that *PSO* is more sensitive to the changing influences, and it can detect the modifications in data more accurately than the *ACO* implementation.

The same conclusion can be drawn when looking at the regression lines. Both lines show a direct influence of the fuzziness amplitude on the solution deviation, but the *PSO* line is closer to the *CPLEX* line showing a better adaptation.

The *PSO* sensitivity is more obvious in Figures 3-5 with the minimum *PE* values taken on instance groups: in each figure, the lines for *PSO* are steeper, showing that this method is more "explorative" than the other, and succeeds in finding new zones in the solution space. The *PSO* behavior shows that it always produces *PE* results with negative deviation, as the exact *CPLEX* method does. We can conclude that *PSO* is able to correctly perceive the data modifications.

The new fuzzy instances when b=25% led to solutions extremely close to the solution of the exact instance. This means that the *PSO* method is able to tolerate data modifications with small amplitude, and to deliver stable results. But there is a clear disparity between the case when a=25% and the other case, with a=10% (the value 0.27% is 9 times higher than 0.03%).

The fuzzy instances derived when b=50% manifest a complete different behavior. The higher amplitude of the data uncertainty provides large variation of the best *PE*, but the couple of cases controlled by the two values for the parameter *a* show a similar result. We can conclude that for large possible intervals of changes of the problem's data, the *PSO* has the same sensitivity, no matter if these changes are few or frequent.

The *Krolak/Felts/Nelson* and the *Padberg/Rinaldi* instances have a distinct and interesting behavior; few changes in node positions (both cases when a=10%) result in larger *PE* values than in the case when more changes are applied (when a=25%). The *Christofides/Eilon* group of the studied instances manifests a direct dependency between the number of fuzzified nodes and the distance of the minimal *PE* from the solution of the exact corresponding instance. We can conclude that the problem structure is very important when a *PSO*-based application is executed. The human decision-maker must also know the problem structure, and does not to entirely rely on the computing results.

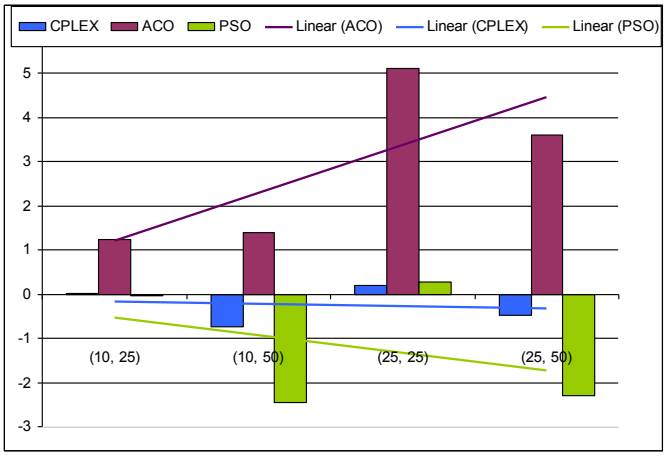

Figure 2. *CPLEX/ACO/PSO* comparison for the overall average *PE in %.*

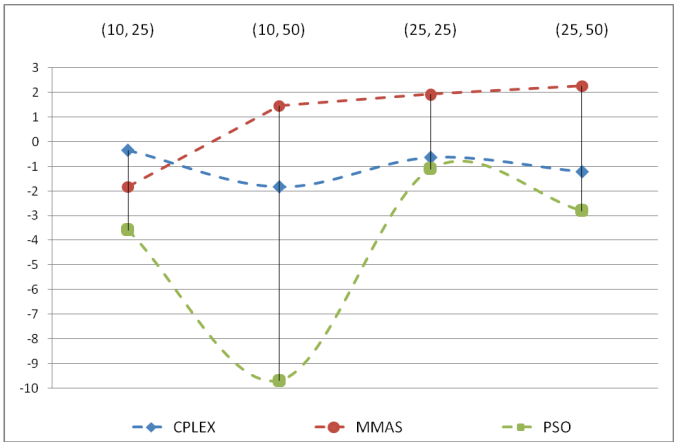

Figure 3. Minimal best *PE* values in % of *CPLEX/ACO/PSO* on the *Krolak/Felts/Nelson* group.

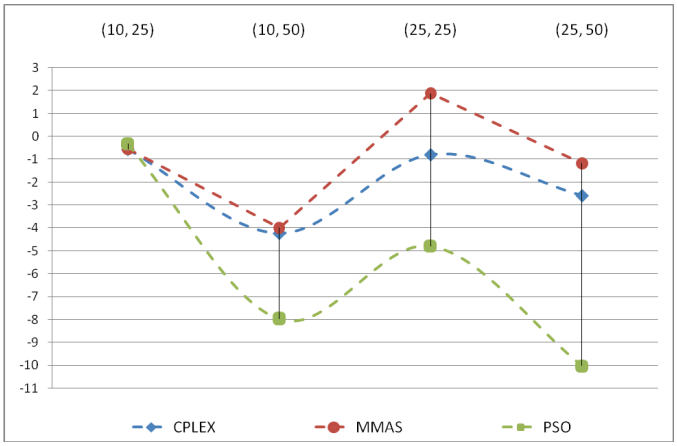

Figure 4. Minimal best *PE* values in % of *CPLEX/ACO/PSO* on the *Christofides/Eilon* group.

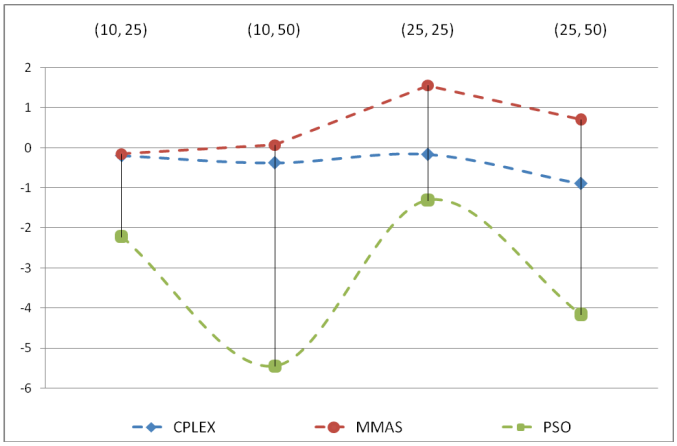

Figure 5. Minimal best *PE* values in % of *CPLEX/ACO/PSO* on the *Padberg/Rinaldi* group.

V. CONCLUSIONS

This paper empirically investigates how a metaheuristic solving method designed for an exact problem performs when it faces a supplementary difficulty level: the data are fuzzified using a two-dimensional uncertainty level: the frequency and the amplitude parameters.

At the practical level, our work shows that the implementation based on *Particle Swarm Optimization* (*PSO*) is stable and fairly adaptable when the uncertainty amplitude is low, but is very sensitive to data changes when this characteristic is high. This means that *PSO* is a reliable choice when almost exact input data are expected. When the data inexactness is unknown, or is expected to be high, then *PSO* can be used as an uncertainty marker: when it provides results further away, one can assume that the data are really far away from the correct ones.

At the theoretical level, this paper uses a new method for introducing uncertainty to a *TSP* instance, through the *fuzzifyInstance* function. It is also a start for other experiments for assessing traditional solving methods when external processes could affect the exactness in data. Other fuzzification

methods, inspired by real-world situations can be tested to assess and compare other solving methods applied to other optimization problems.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable suggestions to improve the paper. The study was conducted under the auspices of the IEEE-CIS-Interdisciplinary Emergent Technologies task force.

REFERENCES

[1] H. E. Stephanou, and A. P. Sage, "Perspectives on imperfect information processing", IEEE Transactions on System, Man, and Cybernetics, SMC-17(5), 780-798, 1987.

[2] B. Russell, The Problems of Philosophy, With a New Introduction by John Perry, Oxford University Press, New York, 1997 edition.

[3] P. P. Bonissone, and R. M. Tong, "Reasoning with uncertainty in expert systems", International Journal on Man-Machine Studies, 22(3), 241-250, 1985.

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

[5] M. M. Balas, and V. E. Balas, "World knowledge for control applications", 11th IEEE International Conference on Intelligent Engineering Systems, pp.225--228, 2007.

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

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

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

[9] I. Finocchi, F. Grandoni, and G. F. Italiano, "Designing reliable algorithms in unreliable memories algorithms", Lecture Notes in Computer Science, vol. 3669, pp.1-8, 2005.

[10] C. C. Aggarwal, and P. S. Yu, "A survey of uncertain data algorithms and applications", IEEE Transactions on Knowledge and Data Engineering, 21 (5), pp.609-623, 2009.

[11] G. C. Crişan, C. M. Pintea, and P. C. Pop, "On the resilience of an ant-based system in fuzzy environments. An empirical study," 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Proceedings pp.2588-2593, 2014.

[12] W. J. Cook, In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press, 2012.

[13] R. M. Karp, "Reducibility among Combinatorial problems", in Complexity of Computer Computations. The IBM Research Symposia, R.E. Miller, J.W. Thatcher (Eds.), pp.85-103, NY: Plenum. Press, 1972.

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

[15] 8th DIMACS Implementation challenge: The Traveler Salesman Problem. Available at: *http://dimacs.rutgers.edu/Challenges/TSP/*

[16] C-M. Pintea, Advances in Bio-inspired Computing for Combinatorial Optimization Problem, Springer, 2014.

[17] C-M. Pintea, C. Chira, D. Dumitrescu, and P. C. Pop, "Sensitive ants in solving the Generalized Vehicle Routing Problem", Int. J. Comput Commun, 6 (4), pp.731-738, 2011.

[18] P. Jaillet, "A priori solution of a Traveling Salesman Problem in which a random set of the customers are visited" Operations Research 36 (6), pp.929-936, 1988.

[19] G. C. Crişan, and E. Nechita, "Solving Fuzzy TSP with Ant Algorithms", International Journal of Computers, Communications and Control, vol. III, suppl. issue, pp.228-231, 2008.

[20] R. Montemanni, J. Barta, and L.M. Gambardella, The robust traveling salesman problem with interval data, Technical Report IDSIA-20-05, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Lugano-Manno, Switzerland, 2005.

[21] Z. C. Huang, X. L. Hu, and S. D. Chen, "Dynamic Traveling Salesman Problem based on Evolutionary Computation", Congress on Evolutionary Computation (CEC'01), IEEE Press, pp.1283–1288, 2001.

[22] C-M. Pintea, P.C. Pop, and D. Dumitrescu, "An Ant-based technique for the Dynamic Generalized Traveling Salesman Problem", Proceedings of the 7-th International Conference on Systems Theory and Scientific Computation, pp.257-261, 2007.

[23] Z. Wang, J. Guo, M. Zheng, and Y. Wang, "Uncertain multiobjective Traveling Salesman Problem", European Journal of Operational Research, 241 (2), pp.478–489, 2015.

[24] G. B. Dantzig, and J. H. Ramser. "The Truck Dispatching Problem", Management Science 6 (1), pp.80–91, 1959.

[25] N. Lahrichi, T. G. Crainic, M. Gendreau, W. Rei, G. C. Crişan, and T. Vidal, "An integrative cooperative search framework for multi-decision-attribute combinatorial optimization", Technical Report CIRRELT-2012-42, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport, Montreal, Canada, 2012.

[26] Concorde solver. Available at: *http://www.math.uwaterloo.ca/tsp/concorde.html*

[27] S. Tschoke, R. Lubling, and B. Monien, "Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network", Proceedings of 9th International Parallel Processing Symposium, pp.182 – 189, 1995.

[28] M. Dorigo, and T. Stützle, Ant Colony Optimization. MIT Press, 2004.

[29] J. Kennedy, and R. C. Eberhart, Swarm Intelligence. Morgan Kaufmann, 2001.

[30] G. Beni, and J. Wang, "Swarm Intelligence in cellular robotic systems", Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, 1989.

[31] R. Poli, "Analysis of the publications on the applications of particle swarm optimization", Journal of Artificial Evolution and Applications, 4, pp.1–10, 2008.

[32] M. Clerc, Discrete particle swarm optimization - illustrated by the traveling salesman problem, New Optimization Techniques in Engineering, Springer, 2004.

[33] J. Q. Chen, W. L. Li, and T. Murata, "Particle swarm optimization for vehicle routing problem with uncertain demand", 4th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp.857 - 860, 2013.

[34] Y. Zhang, D. W. Gong, and J. H. Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization", Neurocomput. 103, pp.172-185, 2013.

[35] P. Novoa, D.A. Pelta, C. Cruz, and I. G. del Amo, "Controlling Particle Trajectories in a Multi-swarm Approach for Dynamic Optimization Problems", International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2009, 5601, Santiago de Compostela, Spain, pp.285-294, 2009.

[36] T. Blackwell, and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments", IEEE Transactions on Evolutionary Computation, 10(4), pp.459-472, 2006.

[37] L. A. Zadeh, "Fuzzy sets" Information and Control 8, 338-353, 1965.

[38] C. V. Negoiţă, and D. A. Ralescu, "Representation theorems for Fuzzy concepts", Kybernetes, 4(3), pp.169-174, 1975.

[39] G. C. Crişan, Ant Algorithms in Artificial Intelligence. PhD Thesis, A.I. Cuza University of Iaşi, Romania, 2008.

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

[41] E. Nechita, C. V. Muraru, and M. Talmaciu, "Mechanisms in social insect societies and their use in Optimization. A case study for trail laying behavior", Proceedings of BICS 2008, 1117(1), pp.171-179, 2008.

[42] M. Clerc, and J. Kennedy, "The Particle Swarm - explosion, stability, and convergence in a multidimensional complex space", IEEE Transactions on Evolutionary Computation, 6(1), pp.58-73, 2002.