The Extension of the Triple Fuzzy Time Dependent Travelling Salesman Problem Model, with a Discrete Bacterial Memetic Optimization Algorithm

Ruba Almahasneh[1], Boldizsar Tuu-Szabo[2] and Laszlo T. Koczy[3]

[1]Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary, mahasnehr@tmit.bme.hu

[2] Department of Information Technology, Széchenyi István University, Gyor, Hungary, tuu.szabo.boldizsar@sze.hu

[3] Department of Information Technology, Széchenyi István University Gyor, Hungary Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary, koczy@tmit.bme.hu

Abstract: The Traveling Salesman Problem (TSP) is one of the most often studied NP-hard graph search problems. There have been numerous publications in the literature that applied various approaches to find the optimum or semi optimum solution. Although the problem is computationally intractable, but the Time Dependent Traveling Salesman Problem (TD TSP) is one of the most realistic extensions of the original TSP problem. In the TD TSP, the costs of edges between nodes vary, namely, they are assigned higher costs if they crossed a predefined oblong shaped area (to represent the jam region in the city center). Realizing that the jam regions and the rush hours costs on a tour are uncertain and can never be accurately represented by concrete numbers, we introduced the novel 3FTD TSP (Triple Fuzzy Time Dependent Traveling Salesman Problem); a fully fuzzified model of the original TD TSP. The 3FTD TSP utilizes fuzzy values for determining the costs between any two nodes within the traffic jam regions and during the rush hours periods more precisely. In this paper, we extend the 3FTD TSP further and apply it on the biggest universal instances in the literature in pursuit of testing the generality and applicability of the 3FTD TSP on real-life scenarios. To support the claim of the model's efficiency, we propose the application of the DBMEA (Discrete Bacterial Memetic Evolutionary Algorithm), as a meta-heuristic and the classic GA (Genetic Algorithm) enabling the reader to compare the accuracy and the speed of (quasi-) optimum solutions convergence

Keywords: Fuzzy Sets; Time Dependent Traveling Salesman Problem; jam region; rush hour period; discrete bacterial memetic evolutionary algorithm

1 Introduction

The Traveling Salesman Problem (TSP) originally attempts to find the minimal overall cost of a route departing from a starting point for a journey and then returning to the starting point, with each location (node) visited only once [1]. In the language of graph theory, it means finding the shortest Hamiltonian cycle. In fact, this problem models many real-life application areas in logistics and transportation route optimization. In the literature, there have been numerous proposed extensions and variations of the original TSP to simulate traffic conditions' added costs (such as traffic regions and rush hours' delays that occur during the day). Those models assigned costs to the edges connecting the nodes (cities) based on Euclidean Distances. However, those costs are constant, which is an obvious limitation when applied to actual cases. Because the traffic circumstances are intangible factors. Moving one step towards a more realistic model, the TD TSP introduced a viable refinement for costs calculation, yet not efficient enough, since it is still a deterministic solution [2]. In fact, the TD TSP assigns a fixed part of the graph to represent the "traffic jam region" in a time dependent form, namely, each graph edge is assigned given costs in the non-rush hour periods, and higher costs during the rush hour times [2]. Afterwards, those crisp numbers are multiplied with the length of the corresponding edge between the nodes in the given rush hour times, to quantify the traffic jam factor's (extra delays caused by traffic such as conjunctions, accidents or rush hours during the day) effects on the respective costs [2]. Despite the TD TSP's ability to achieve good results in determining the overall cost, one main drawback is the use of concrete numbers for representing the proportional traffic jam factors and the simplified representation of the traffic jam region (city center). After the classical TD TSP, there have been many other attempts by researchers to quantify the rush hours and the traffic jam regions effects on the trip's overall cost. In 2015, Hurkała proposed a new algorithm for the TD TSP with multiple time windows and compared it with three well-known and often efficient heuristic methods (simulated annealing, list-based threshold accepting algorithm, and variable neighborhood search) [3]. The main drawback of these results is that the methods mentioned were tested on small instances, up to 23 nodes only. In 2016, Taş et al. proposed a variant of the Traveling Salesman Problem with time-dependent service times [4]. The mathematical formulation of the model was extended with sub-tour elimination constraints and was solved by using IBM ILOG CPLEX 12.5. The tests were carried out only on smaller instances up to 45 nodes. The tests were carried out only on smaller instances up to 45 nodes. Even for these relatively small graphs, in quite many cases the optimal solutions were not found within the time limit set by the authors (7200 s). In 2019, Vu et al. presented an integer programming formulation for solving the TD TSP with time windows. However, this approach was again only able to solve small instances, up to 40 nodes [5]. In 2018, our team [6] also published a novel (non-fuzzy) approach, where a parameter-adapted version of the rather generally applicable (say, "universal"), efficient, and efficacious method called the Discrete Bacterial Memetic Evolutionary Algorithm (DBMEA), was applied for solving the problem, first, for the reason of possible comparison, on Schneider's instances. The proposed meta-heuristics clearly outperformed previous attempts for the solution of this problem, and, by adding several new, larger instances, it was also shown that DBMEA did not have restrictions concerning the size of the graph (rather, when knowing the size, a good prediction of obtaining a good approximate solution could be obtained).

In 2019, Ban proposed a two-phase meta-heuristic algorithm [7] for Schneider's version of the Time-Dependent Traveling Salesman Problem, but it did not outperform the earlier published DBMEA approach [2]. Moreover, some models used Ant Colony optimization to predict traffic for the TD TSP, where the traveled time between cities changes with time [9]. Others used a mixed approach, where they incorporated integer programming model to calculate the traveled time, then integrated both Ant Colony System (ACS) and Tabu Search (TS) algorithms [10] to calculate the overall cost. Other authors used linear constrains formulation techniques to present the traffic jam factors' effect on any trip [11]. Also, dynamic programming algorithm is proposed to solve the TD TSP [2]. Looking closely, there is one common aspect in all the previously proposed approaches, that is, using crisp numbers to simulate those uncertain and imprecise road conditions, indeed, this is an insufficient formulation [11] [12]. Conversely, there was one attempt to apply fuzzy sets on the TSP in 2017; the fuzzy based solution to the travelling salesman problem introduced [13], this model was applied on 5 nodes (cities) and used triangular membership functions to represent all traffic conditions as added costs to the overall edges' length. The drawback of this representation is obvious, the model's efficiency, practicality or applicability was never proven or tested on large scale instances. In fact, it would be closer to reality to use fuzzy numbers to capture the uncertainties associated with road conditions (traffic jam factors and rush hours) while simulating actual cases for courier trips. Consequently, we introduced three novel fuzzy-based novel approaches (3FTD TSP, IFTD TSP, IVIFTD TSP) which offered for the very first time a tangible and realistic formulations for the jam regions and the rush hours periods costs.

***First model*:** The Triple Fuzzy Time Dependent Travelling Salesman Problem (3FTD TSP) model [14]. It is called triple fuzzy as it has three aspects of uncertainty that are represented by type-1 fuzzy sets (characterized by two-dimensional membership functions). In the 3FTD TSP, the costs between the nodes are expressed by fuzzy sets, which best suited for non-deterministic factors, such as unexpected accidents, road constructions, weather conditions, etc. Here, the fuzzy values represent the vagueness and imprecision of the traffic factors which result in additional costs on the tour. In the example of the paper [14], the rush hours times were represented as bimodal piecewise linear normal fuzzy sets. On the other hand, the jam areas were represented by fuzzy trapezoidal "oblong shaped" sets of the affected areas. This model calculated the overall tour length (cost) more efficiently.

***Second model*:** The Intuitionistic Fuzzy Time Dependent Travelling Salesman Problem (IFTD TSP) model [15]. An even more general model that allowed road uncertainties representation to become more flexible. This model employs type-2 Intuitionistic fuzzy sets (IFSs) which are characterized by three-dimensional membership functions. The IFSs was introduced by Atanassov [16] [17] which is the core of this model. IFSs involve the so-called hesitation and the non-membership part between the membership functions. IFSs theory has been applied in different areas that have to do with decision making under high hesitation and vagueness degrees and it proved being more adequate [18] [19] than the classic crisp representations. Thus, the use of intuitionistic fuzzy sets in the IFTD TSP model indeed ensured successful representation for higher degrees of association and the lower degrees of non-association for the traffic jam factors and the rush hours and lower degrees of hesitation to edges' costs in any proposed tour, that resulted in a more accurate overall cost estimation for any given tour.

***Third model*:** In pursuit of considering the accumulative average costs of multiple factors affecting a single edge simultaneously; we proposed the interval-valued intuitionistic fuzzy Time Dependent Travelling Salesman Problem model (IVIFTD TSP) [20]. This model also uses the interval-valued intuitionistic fuzzy sets (type-2). The improvement we achieved in this model was employing the aggregation concept of all the costs rather than using the max-min composition of the fuzzy factors [15] (as in the IFTD TSP). Since in an actual tour, any edge can be affected by several factors simultaneously, and thus, by taking in consideration all the external traffic factors affecting any edge collectively at any given time, less information loss was guaranteed [20]. Indeed, this formulation led to an even more adequate model.

All three models [21] were tested on small instances and proven efficient, continuing our work, in this paper we extended the 3FTD TSP on bier127, eil51, eil76, eil101, bier127 and our own instances which consist of 250 nodes (s250_1, s250_2); those universal instances are the largest extended TSP benchmark in the literature. Our aim is proving the 3FTD TSP's efficiency, generality and practicality on real-life scenarios.

Generally, Memetic algorithms have been applied in various fields and they have been proven efficient [22]. Calculating the modified tour costs is more complex and time consuming. It requires even more operational power, hence, the GA (Genetic Algorithm) and DBMEA (Discrete Bacterial Memetic Evolutionary Algorithm) [23] [24] were used to optimize the (quasi-) optimum or semi optimum solution to achieve best possible convergent speed.

2 Formulation of the Classic TSP Solution

The original TSP was first formulated in 1930 [1], it involves a salesman who starts the journey from the company headquarters, visits all planned destination (cities or shops) exactly once, and then returns to the original starting point with the minimum overall travelled distance. TSP can be defined as a graph search problem with edge weights (costs) as in Eqs. (1, 2, 3 and 4). To formulate the symmetric case with n nodes (cities). Let $c_{ij} = c_{ji}$represents the cost of going from city (vertex) i to city j. G_{TSP} is the graph on hand, then, C is called cost (time) matrix, and can be expressed as:

$$C = [c_{ij}] \tag{1}$$

G_{TSP} graphisa combination of a list of vertices and edges such that $V_{cities}$is the set of all vertices between the cities. E_{conn} is the set of all edges in the graph, and $v_1$is the starting vertex. Then:

$$G_{TSP} = \langle V_{cities}, E_{conn} \rangle \tag{2}$$

thusV_{cities} can be expressed as:

$$\mathrm{V_{cities}} = \mathrm{v_1} \cup \{\mathrm{v_2}, \dots \mathrm{v_i} \dots., \mathrm{v_n}\}, i = (2 \dots, n) \tag{3}$$

and

$$E_{conn} \subseteq \{(v_i, v_j) | i \neq j\}, \mathrm{i}, \mathrm{j} = (1 \dots, \mathrm{n}) \tag{4}$$

The cost matrix C is the mapping of all subscripts of the vertices, can be expressed as:

$$C: V_{cities} \times V_{cities} \to R^{+} \tag{5}$$

$R^{+}$is the range of the costs from the set of edges to the positive real lines. The initial set of vertices in any tour is $\{1, k_2, \dots k_i \dots., k_n\}$ where $k_1 = 1$ the starting subscript, $i = (2 \dots, n)$. The goal is to find the path of edges that minimizes the sum of the costs (C_{sum}) by traveling through the vertices exactly once. C_{sum} calculated in Eq. (6) and should be minimized as in Eq. (7). Here c_{1,k_2} is the cost of traveling the starting edge, and $c_{k_{n-1},1}$ is the cost of returning to the first vertex (traveling the last edge).

Thus, C_{sum} can be written as:

$$C_{sum} = c_{1,k_2} + \sum_{j=2}^{n} c_{k_j,k_{j+1}}, (j = 2, \dots, n-1)$$

$$, c_{k_j,k_{j+1}} \epsilon [c_{ij}], k_2, k_n \epsilon \{2, \dots, n\} \tag{6}$$

Then to minimizes the sum of the costs:

$$\text{Minimize } C_{sum} \tag{7}$$

A valid permutation for vertices (the subscripts) is:

$$\{v_1, v_{k_2}, \dots v_{k_i} \dots, v_{k_n}\} \tag{8}$$

where $i = (2 \dots, n).v_1$ is the starting vertex.

Let $\{k_1, \dots . k_n\}$ be a permutation of $\{1 \dots, n\}$, then a valid sequence of edges that forms a path that goes through all vertices (exactly once and goes back to the first one) can be expressed in the form:

$$e_{1,k_2}, e_{k_2,k_3}, \dots . e_{k_i,k_j} \dots \dots e_{k_{n-1},1} \text{ , } i, j \in \{2 \dots, n\} \tag{9}$$

where the first edge is e_{1k_2} , $k_i \neq k_j$ so that $i \neq j$.

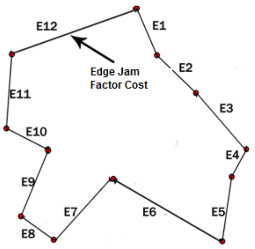

Figure1
Tour for a Simple Example

Considering an actual trip (for simplicity we will consider small sized tour Fig. 1), clearly, it is subjected not only to the topography of the given route, but also to the unexpected circumstances and road conditions, which affect the overall timing (cost) of the tour. Naturally, in the city centers and during the rush hours periods, trips between nodes (vertices) take longer. Thus, those two factors must be looked at as vague or uncertain (non-deterministic but not in a random sense) values, and the relevant estimated costs for any tour traveled between two nodes are not constant themselves. Hence, it is more appropriate to represent those imprecise (uncertain) costs using fuzzy numbers. This was my main motivation for introducing the novel 3FTD TSP model, which is a more adequate and realistic version of the classic TD TSP (the crisp version fuzzy-based from multiple aspects).

3 The Triple Fuzzy Time Dependent Traveling Salesman Problem (3FTD TSP)

The 3FTD TSP is so far the only fully extended and published fuzzy version of the classic TD TSP using type-1 fuzzy sets [25]. In the 3FTD TSP model, three “modifying parameters” of the edge costs were represented using fuzzy sets. Each cost in the trip has its own membership function, namely, the fuzzy time (cost) needed for crossing edge$(v_i , v_j)$is denoted by:

$$c(v_i, v_j) = \tilde{c}_{ij} \tag{10}$$

The fuzzy jam region influenced cost with membership function $\mu_J(v_i, v_j)$ and the fuzzy rush hours cost with membership function $\mu_{RH}(h)$ (See Fig. 2 and Fig. 3). Hence, according to the degree of belonging to the specified jam region or if the edge was being crossed during the fuzzy rush hours; the original fuzzified time of the related edge(s) will be influenced accordingly. It is worth mentioning that, in our fuzzy formulation, the cost of crossing the edges is defined by the time needed to cross them. The cost between edges is expressed by time, thus, from here on we will refer to time as cost. Moreover, cost here is a fuzzy set of the universe of the potential costs between two vertices (the universe of potential travelling time).

Let c_{ij} be the cost of traveling between vertices$v_i$andv_j, $\tilde{C}$ is the fuzzy cost matrix, which can be defined as:

$$\tilde{C} = [\tilde{c}_{ij}] \tag{11}$$

where $\tilde{c}_{ij}: T \rightarrow [0,1]. i \neq j$, and $i, j \in \{1, \dots . n\}$

T is the universe of the costs $[0, \max\{T\}]$, such that $(\max\{T\})$ is the highest cost in the trip under study.

Thus,

$$\tilde{c}(v_i, v_j) = \tilde{c}_{ij} \tag{12}$$

$$\tilde{c}_{ij} = \langle T, \mu_{ij}(t): T \rightarrow [0,1] \rangle, \text{ such that } t \in T \tag{13}$$

In our illustrative example, $\mu_{ij}(t)$ defines $\tilde{c}_{ij}$ over T (the traveling time between vertices $v_i$andv_j). $\mu_{ij}(t)$ is a triangular membership function between vertices $v_i$andv_j. $x, y,$ and z represent the break points of $\langle \mu_{ij}(x, y, z) \rangle$ in Eq.(14):

$$\tilde{c}(v_i, v_j) = \langle T, \mu_{ij}(x, y, z) \rangle \tag{14}$$

$$\mu_{ij}(x, y, z) = \begin{cases} 0 & (v_i, v_j) \leq x \\ \frac{(v_i, v_j) - x}{y - x} & x \leq (v_i, v_j) \leq y \\ \frac{z - (v_i, v_j)}{z - y} & y \leq (v_i, v_j) \leq z \\ 0 & z \leq (v_i, v_j) \end{cases}$$

$$= \max\left\{\min\left\{\frac{(v_i, v_j) - x}{y - x}, \frac{z - (v_i, v_j)}{z - y}\right\}, 0\right\} \tag{15}$$

To calculate the modified cost due to crossing the jam region, we assume (in the presented example) there is a center point at the city center, and the jam region has a truncated conic shaped membership function $\mu_J(v)$. Where the membership degree depends on the distance measured from the center point. Since we assume a truncated conic type membership function for the jam region $\langle x, y; r1, r2 \rangle$, there are two radiuses ($r1$ and $r2$); one that of the conical kernel ($r1$) and the other is that of the support ($r2$). To calculate the modified cost that affects edge (v_i,v_j) due to crossing the jam region, we must first find the membership degree μ_J of each vertex. Let V be set of vertices that is $V = v_1 \cup \{v_2, \dots v_i \dots., v_n\}, v_1$ is the starting vertex. $i = (2, \dots n).J$ denotes the fuzzy set of the jam region.

$$J = \langle V, \mu_J(v): \to \quad [0,1]\rangle \qquad , \forall\, v \in V \tag{16}$$

Then, to calculate the cost modified by being (partially) in the jam region for edge (v_i, v_j), we apply Eq. (17):

$$\mu_{CJ}(v_i, v_j) = \frac{\mu_J(v_i)+\mu_J(v_j)}{2} \tag{17}$$

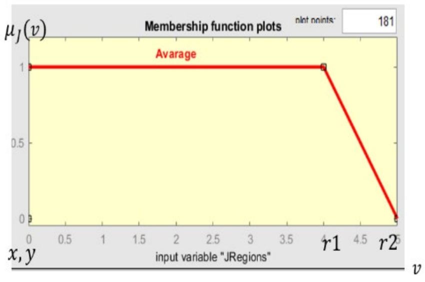

Figure 2
One dimensional cut of ($\boldsymbol{\mu_J}$)

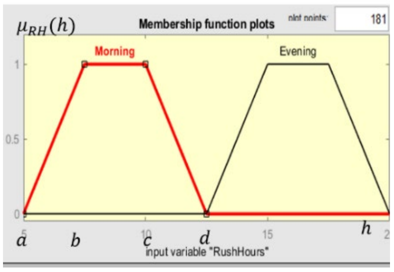

Figure 3
Traffic rush hours' for (μ_{RH})

Fig. 2 shows a graphic example cut of the jam region membership function in two dimensions, having a trapezoidal shape with break point values of [0, 0, 4000, 5000]. We use Eq. (18), which defines a trapezoidal membership function $(v; \dot{a}, \dot{b}, \dot{c}, \dot{d})$, with $\dot{a}, \dot{b}, \dot{c}$ and $\dot{d}$ representing the break points. Where, $\dot{a} = \dot{b} = \dot{c}$ (in our example).

$$\mu_J(v; \dot{a}, \dot{b}, \dot{c}) = \begin{cases} 1 & v \le \dot{c} \\ \frac{\dot{d}-v}{\dot{d}-\dot{c}} & \dot{c} \le v \le \dot{d} \\ 0 & \dot{d} \le v \end{cases} \tag{18}$$

This formulation eliminates (to some extent) one of Schneider's model inaccuracies, which ignores the intersections of the jam regions and the fact that such modified cost builds up gradually (which is better captured using fuzzy sets as we have shown).

To show the effect of the modified fuzzy cost that might affect a trip is the rush hour periods. We propose a double trapezoidal membership function for the traffic rush hour periods during the day (in our simple illustrative example, see Fig. 3). It has the break point values {5 ,7.5, 10, 12.5, 12.5, 15, 17.5, 20}, and its membership values are: {0, 1, 1, 0, 0, 1, 1, 0, 0}. $\mu_{RH}(h)$ was represented with two main rush hour periods during the day (with membership values equal to 1) the peaks are from 7.5 a.m. to 10 a.m. and from 3 p.m. to 5.5 p.m. Between the two periods the traffic intensity is lower but not zero. Let H be the hours' time line (24 hours of the day), h the elements of the time line such as $h \in H$.

The membership function for the rush hours is:

$$RH = \{H, \mu_{RH}(h)\}, \mu_{RH}(h): H \rightarrow [0,1], h \in H \tag{19}$$

$\mu_{RH}(h)$is the degree of membership of belonging to the rush hour periods (whether the edge was crossed during rush hours periods). Here a, b, c and d represent the break points, then the a trapezoidal ($h;\ a, b, c, d$) membership function for the rush hours periods (in the morning or the evening in our example) can be written as in Eq. (20):

$$\mu_{RH}(h) = \begin{cases} 0 & h \le a \\ \frac{h-a}{b-a} & a \le h \le b \\ 1 & b \le h \le c \\ \frac{d-h}{d-c} & c \le h \le d \\ 0 & d \le h \end{cases} \tag{20}$$

$$\mu_{RH}(h) = \max\left\{\min\left\{\frac{h-a}{b-a}, 1, \frac{d-h}{d-c}\right\}, 0\right\}$$

Finally, to calculate the resulting fuzzy cost that can possibility affect a trip (all edges in a trip), we must first sum the resulting fuzzy costs on each edge (v_i, v_j), then on the whole trip. Hence, for calculating the resulting fuzzy cost $\tilde{C}_{sum}$ between vertex v_i and v_j we apply Eq. (21).

$$\tilde{C}_{sum}(v_i, v_j) = [\tilde{c}(v_i, v_j)] * [1 + \mu_{RH}(h)] * \left[1 + \frac{\mu_J(v_i) + \mu_J(v_j)}{2}\right] \tag{21}$$

Let $\{k_1, \dots . k_n\}$be a permutation of $\{1 \dots, n\}$, then a valid sequence of edges that forms a path that goes through all vertices (exactly once and goes back to the first one) can be expressed in the form:

$$p = e_1, e_{k_{2,}k_3}, \dots . e_{k_{i,}k_j} \dots \dots, e_1 \text{ , } i \neq j \text{ so that} k_i \neq k_j \text{and } i, j \in \{1 \dots, n\} \tag{22}$$

where the first edge is e_1.

To calculate the cumulative modified fuzzy cost for the first two edges e_1(with membership break points a, b, c) and $e_{k_{2,}k_3}$ (with membership break points p, q, r) whose triangular membership functions are shown in Eq. (13), we apply Eq. (23) [87]:

$$\bar{\bar{\boldsymbol{C}}}_{\boldsymbol{sum}}(\boldsymbol{e_1} + \boldsymbol{e_{k_{2,}k_3}}) = \boldsymbol{\mu_{e_1 + e_{k_{2,}k_3}}(t)}$$

$$= \begin{cases} \frac{\boldsymbol{t}-(\boldsymbol{a}+\boldsymbol{p})}{(\boldsymbol{b}+\boldsymbol{q})-(\boldsymbol{a}+\boldsymbol{p})}, & (\boldsymbol{a}+\boldsymbol{p}) \le \boldsymbol{t} \le (\boldsymbol{b}+\boldsymbol{q}) \\ \frac{(\boldsymbol{c}+\boldsymbol{r})-\boldsymbol{t}}{(\boldsymbol{c}+\boldsymbol{r})-(\boldsymbol{b}+\boldsymbol{q})}, & (\boldsymbol{b}+\boldsymbol{q}) \le \boldsymbol{t} \le (\boldsymbol{c}+\boldsymbol{r}) \end{cases} \tag{23}$$

We apply Eq. (24) to all edges of the path (p) to find $\bar{\bar{C}}_{sum}(p)$.

$$\bar{\bar{\boldsymbol{C}}}_{sum}(\boldsymbol{p}) = \begin{cases} \frac{t-(\sum_{ij\in n} a_{ij})}{(\sum_{ij\in n} b_{ij})-(\sum_{ij\in n} a_{ij})}, & (\sum_{ij\in n} \boldsymbol{a_{ij}}) \leq \boldsymbol{t} \leq (\sum_{ij\in n} \boldsymbol{b_{ij}}) \\ \frac{(\sum_{ij\in n} c_{ij})-t}{(\sum_{ij\in n} c_{ij})-(\sum_{ij\in n} b_{ij})}, & (\sum_{ij\in n} \boldsymbol{b_{ij}}) \leq \boldsymbol{t} \leq (\sum_{ij\in n} \boldsymbol{c_{ij}}) \end{cases} \tag{24}$$

In Fig. 2 and Fig. 3, the following three elements are described by fuzzy membership functions [zero values here mean the normal traffic, away from the city center and not during rush hours]:

a) Triangular fuzzy costs (time) between the edges;$\mu_{ij}(t)$

b) Membership function of the fuzzy jam region; $\mu_{CJ}(v_i, v_j)$

c) Membership function(s) of the traffic rush hour time period(s) $\mu_{RH}(h)$

4 The Defuzzification in the 3FTD TSP

Defuzzification is the process of obtaining a single number from the output of the aggregated fuzzy set. It is used to transfer fuzzy inference results into a crisp output [26]. We used the center of gravity (COG) [27], which is one of the most commonly used defuzzification methods (other defuzzification methods are; mean of maximum (MOM) [28-30], and center average methods, the center of area method (COA) [27]). Furthermore, the weighted average method [29] [30] for finding the crisp output value, y^* is computed by taking the sum of the multiplication of each weighting functionμ_y, with the maximum value of its respective membership value $\bar{y}$, and dividing it by the sum of the weighting functions. This representation was chosen when I implemented the 3FTD TSP C++ modules due to computational complexity (calculation of COA and weighted average method show very close results). This concept is presented in Eq.(25):

$$y^* = \frac{\sum_{i=1}^{n-1}[\mu_y(\bar{y})\times\bar{y}]}{\sum_{i=1}^{n-1}\mu_y(\bar{y})} \tag{25}$$

Finally, these crisp numbers $y^*$are summed up providing the total distance of the tour.

4.1 The Fuzzy Inference Rules for the 3FTD TSP

The fuzzy rules and output membership functions for our model are shown in Fig. 4. The fuzzy rules in the 3FTD TSP are implemented as below:

1) *IF* JamRegion is Average AND Rush is Morning *THEN* Signal is HCost
2) *IF* JamRegion is Average AND Rush is Evening *THEN* Signal is MCost
3) *IF* JamRegion NOT Average AND Rush is Morning *THEN* Signal is LCost

4) *IF* JamRegion NOT Average AND Rush is Evening *THEN* Signal is LCost
5) *IF* JamRegion is Average AND Rush NOT Morning *THEN* Signal is MCost
6) *IF* JamRegion is Average AND Rush NOT Evening *THEN* Signal is MCost

5 The Genetic Algorithm and the DBMEA

Genetic Algorithms (GA) are the archetypical evolutionary search and optimization algorithms that are based on concepts of natural selection and natural genetics. The original GA algorithm was developed to simulate some of the processes observed in natural evolution, a process that operates on chromosomes. It searches among a population (of points) and uses objective function information without any gradient information [31] [32]. The transition scheme of the GAs is used as general-purpose optimization algorithm. GAs are also efficient to search in irregular spaces, and thus, are applied to a variety of function optimizations and parameters estimation problems [31]. On the other hand, the DBMEA is a memetic meta-heuristic algorithm, it is an extension of the original Bacterial Evolutionary Algorithm (BEA), which was proposed as a further development of the GA [32]. The Memetic algorithms [33] get their efficiency from the fact that they combine both global search and a local search, the first being slower but gives a guarantee of finding the quasi-optimum, while the second being very fast and precise but may not find the real optimum [33-35]. The DBMEA is based on the BEA method proposed by Nawa and Furuhashi [23], combined with local search techniques which combination eliminates the disadvantages of both methods. This formulation significantly improved the performance of the classical evolutionary algorithms, so the memetic algorithms can be considered as efficient tools for solving the TD TSP and many other NP- hard optimization problems [1]. In the 3FTD TSP, in every iteration, a local search 2-opt step is applied for the tour, and it is stopped when the tour stops improving. Afterwards, the 3-opt step is applied on the so-far best tour, until there is no further enhancement plausible [35].

6 Computational Results

Since we first introduced the 3FTD TSP, we were eager to test it on the extensions of the largest universal TD TSP benchmarks, to prove its efficiency and compare it with state-of-the-art solutions in the literature. Thus, we have developed the 3FTD TSP model using C++ to be able to represent the jam regions and rush hours' fuzzy extensions [27] [28] and integrate it with the DBMEA and so far, one other optimization algorithm (GA) for comparison purposes. The benchmarks we used are: eil51, bier127, eil76, eil101 s250_1 and s250_2. Each instance's name reflects the number of cities (nodes) under study. To illustrate, Eil51 has a total of 51 cities that a salesman must visit. The coordinates of all the cities have been downloaded from the library TSP-Lib12. We also used our own instances (s250_1 and s250_2) which consist of 250 cities (each). In addition, we have successfully applied the DBMEA algorithm and the GA on our proposed model for optimization to allow the reader to compare the results from different aspects, the speed, the efficiency and generality. Table 1. and Table 2. show different run results for the Bier127 benchmark problem applied on the extended 3FTD TSP, with different traffic jam factors. This original TSP benchmark is modelling the locations of the Bier127 gardens in Augsburg (Germany). The traffic jam region was defined with a circle of radius 5000m and with center (10540, 11945). Velocity v is 6000 m/h in each test. The middle point of the fuzzy triangular cost for each edge is the Euclidean distance between the endpoints, namely, the left-side and right-side points were determined randomly (0-50% lower and higher than the middle point) in this test. Our algorithm was tested on an Intel Core i7-7500U 2.7 GHz, 8GB of RAM memory workstation under Linux Mint 18. The results were calculated by averaging five test runs (Table 1). The Table contains the total time in hours required to visit each location with different jam regions and rush hours costs [26-28].

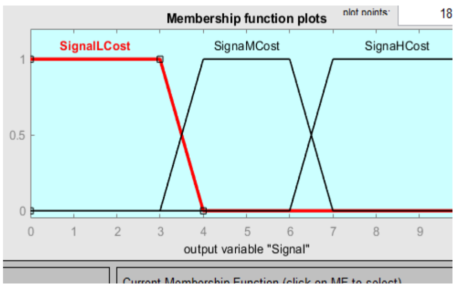

Figure 4
Output costs membership function

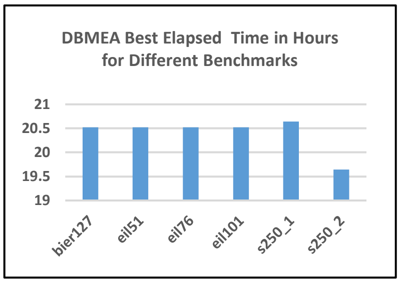

Figure 5
Best elapsed time in hours for different benchmarks with DBMEA

Table 3 shows different run results for the Eil51 benchmark problem applied using the fuzzified version (3FTD TSP), with different jam regions and rush hours' values. Table 4. shows different run results for the Eil76 benchmark problem applied using the 3FTD TSP with different jam regions and rush hours' values. Table 5. show different run results for the Eil101 benchmark problem applied using 3FTD TSP with different jam regions and rush hours' values. Table 6. shows different run results for the s250 benchmark problem applied using the 3FTD TSP with different jam regions and rush hours' values. Table 7. shows different run results for the s250 benchmark problem applied using the 3FTD TSP with different jam regions and rush hours' values.

Table 1
computational results for 3FTD TSP with DBMEA

Testing Instance	Best elapsed time			Best elapsed time	Average elapsed time	Average runtime [s]
bier127	20.5317273	20.58075	20.52149	20.52149	20.54466	5.698
s250_1	21.08152503	20.71037	20.64352	20.64352	20.8118	27.493
s250_2	19.75256584	19.67563	19.64307	19.64307	19.69042	27.383

Testing our algorithm on standard TD TSP benchmarks and achieving promising results (Tables 1 to 10, Figs. 7 and 8) only proves that our new model is general enough to be applied for the extended problem, and we also want to show that it converges with a (quasi-)optimal solution in all cases. Indeed, the DBMEA (Figs. 5 and 6) was able to find high quality solutions for the 3FTD TSP problem, with different membership functions *J* and *R*, even with large rush hours and jam regions values.

In fact, this proves that the 3FTD TSP, namely, our extension of the TD TSP with fuzzy sets gives realistic run times and a timely conversion speed which proves the applicability of the model. In addition, it indicates that the 3FTD TSP is a closer, more general and robust solution to simulating real life scenarios than the original TD TSP is, the former offering more realistic representation of the rush hours and jam regions with high adequacy. It is worth mentioning here that having close conversion times (for both the best and average times in Figs. 5, 6, 7, 8) only proves that the DBMEA was consistent in its performance in finding the (quasi-)optimal solution repeatedly, thus, the novel proposed model can (to some extent) safely be generalized.

The obtained results are validated in the context of the known optimum solution for the original TSP, and Schneider's [2] solution for the TD TSP. Similar results were obtained in our previous work [7] (Table 1) for smaller instances. When the loss aversion approach was applied and the supports rather than the cores or the whole membership functions of the fuzzy cost factors were taken into account (e.g., with traffic jam factor = 5.0) the average elapsed time was 22.286 hours, the average support value of the output triangular fuzzy numbers was 10.76 hours when the loss aversion was taken into account (without loss aversion, these values were 22.196 hours and 10.92 hours).

Table 2
Bier127 computational results for 3FTD TSP with DBMEA

	Elapsed time			Best	Average elapsed time
bier127	20.5317273	20.5807	20.5215	20.5215	20.54465592
bier127	20.098	20.399	20.24915	20.098	20.249
s250_1	21.08152503	20.71037	20.64352	20.64352184	20.81180495
s250_2	19.75256584	19.67563	19.64307	19.64307403	19.69042

Table 3
Eil51 computational results for 3FTD TSP with DBMEA

Eil51	**elapsed time**			**best time**	**Average time**
	15.7147	15.7573	15.7147	20.5215	15.7289
51-city problem (Christofides/Eilon) TYPE: TSP					

Table 4
Eil76 computational results for 3FTD TSP with DBMEA

Eil76-hours	**elapsed time**			**elapsed time**	**average elapsed times**
	21.4841	21.5433	21.7321	20.5215	21.5865
eil76.tsp - 76-city problem (Christofides/Eilon)					

Table 5
Eil101 computational results for 3FTD TSP with DBMEA

Eil101 -hours	**elapsed time**			**best**	**average elapsed time**
	23.6234	23.5858	23.5537	20.5215	23.5876
101-city problem (Christofides/Eilon)					

Table 6
s250_1 computational results for 3FTD TSP with DBMEA

s250_1 -hours	**elapsed time**			**best**	**average elapsed time**
	21.08152503	20.7104	20.6435	20.6435	20.8118
250-city					

Table 7
s250_2 computational results for 3FTD TSP with DBMEA

s250_02 - hours	**elapsed time**			**best time**	**average elapsed time**
	19.75256584	19.6756	19.6431	19.6431	19.6904
250-city					

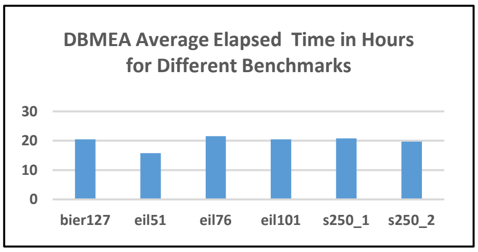

Figure 6
Average elapsed time in hours for different benchmarks with DBMEA

Table 8
bier127 computational results for 3FTD TSP with DBMEA

	DBMEA 3FTD TSP	**Classic TD TSP**
jam factor	**best value**	**best value**
1.00	115929.0	118293.5
1.05	117498.6	119053.2
1.20	117997.2	119714.1
1.50	121076.4	120571.7
2.00	125083.8	121125.2
3.00	129369.0	121125.2
5.00	132220.2	121125.2
10.00	137813.4	121125.2
20.00	139286.4	121125.2
50.00	144363.1	121125.2
100.00	147520.2	121125.2

Table 9
100 Generation comparison of results with DBMEA and Genetic Algorithm for the 3 FTD TSP

	DBMEA		**Genetic Algorithm**	
Benchmark	**best value**	**average value**	**best value**	**average value**
eil51	**15.715**	**15.729**	15.744	15.918
eil76	**21.484**	**21.587**	21.553	21.919
eil101	**23.554**	**23.588**	23.580	23.832
bier127	**20.048**	**20.296**	20.611	21.189
s250_1	**20.644**	**20.812**	21.487	21.587
s250_2	**19.643**	**19.690**	20.132	20.177

We have run both algorithms on 300 generations. The Genetic Algorithm was faster than DBMEA after 100 generations. However, what makes DBMEA generally outperform the GA is the fact that, to some surprise, the GA gets stuck in a local optimum and seems unable to significantly improve the solution between 100 and

300 generations. Of course, there may be some changes after a larger number of generations, but this could result in infeasible runtime. In Fig. 7, the results show the comparison of the convergence times (best times) for both the GA and DBMEA using the 3FTD TSP. It is clear that in most of the runs, DBMEA outperformed the GA. In Fig. 8, the graph compares the average runtimes for both algorithms. The GA is slightly faster initially, but as the problem size increases, DBMEA shows faster convergence, while the GA tends to get stuck in local optima. For instance, in the case of the 250-point problems (s250_1 and s250_2), DBMEA performs better both in the best value and average value. Specifically, for s250_1, DBMEA has a best value of 20.644 and an average value of 20.812, compared to the GA's best value of 21.487 and average value of 21.587. For s250_2, DBMEA achieves a best value of 19.643 and an average value of 19.690, while the GA's best value is 20.132 and the average value is 20.177. These results show DBMEA performing approximately 3.9% and 2.6% better than the GA on average for s250_1 and s250_2, respectively.

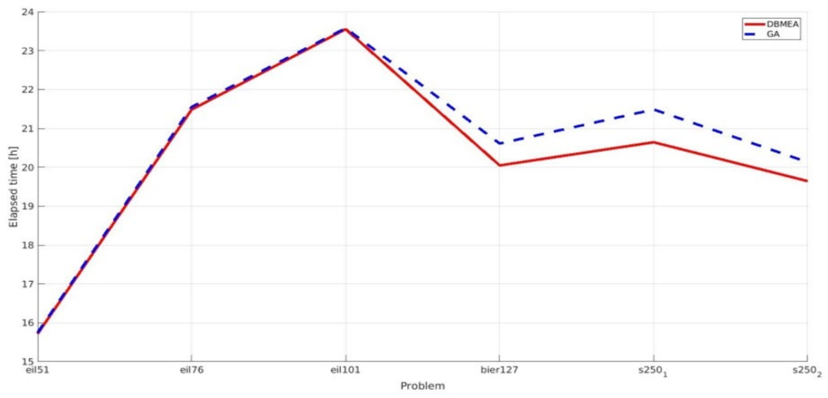

Figure 7
DBMEA vs. GA, comparison of best tour values of universal benchmarks after 100 iterations

For the benchmark problem ier127, statistical tests were conducted over 20 runs to compare the performance of DBMEA and the Genetic Algorithm (GA). Figure 9 illustrates the performance comparison between the two algorithms over 100 iterations. The shaded regions represent the min-max intervals, while the solid lines indicate the average performance for each algorithm. It is evident that DBMEA consistently outperforms GA throughout the iterations. By the 100th iteration, the worst result achieved by DBMEA is approximately equal to the best result of GA, highlighting DBMEA's superior convergence and robustness. Moreover, after just 10 iterations, the average result of DBMEA exceeds the average result of GA after 100 iterations. The Shapiro-Wilk test for normality showed that both algorithms' results follow a normal distribution, with p-values for both DBMEA and GA being above the 0.01 threshold, indicating no significant departure from normality (Table 10).

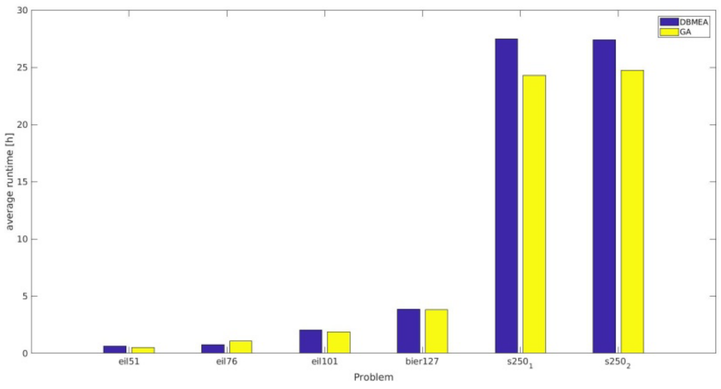

Figure 8
DBMEA vs. GA for average runtime comparison of universal benchmarks after 100 iterations

To determine whether the differences in performance between the two algorithms were statistically significant, we conducted Welch's t-test. The results indicated a significant difference, with DBMEA outperforming GA (Table 11). This finding underscores the superior performance of DBMEA in comparison to GA for the bier127 benchmark, with the difference being statistically significant, at the 1% level.

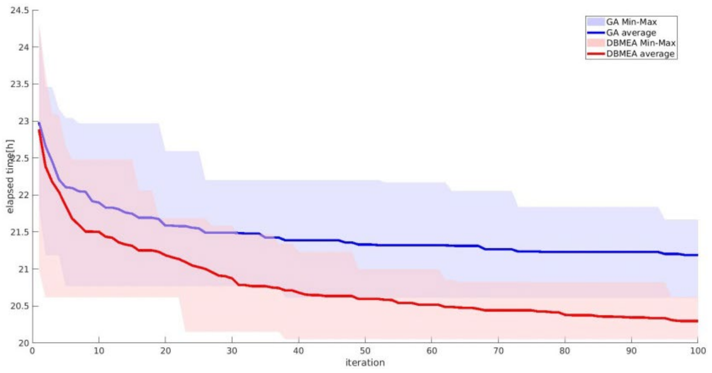

Figure 9
GA vs. DBMEA performance over Iterations

Table 10
Statistical Analysis of Results for the bier127 Benchmark

Benchmark	Algorithm	Best value	Average value	Standard deviation	Shapiro-Wilk p-value	Shapiro-Wilk W	Hypothesis H_0
bier127	DBMEA	20.048	20.296	0.184	0.185	0.934	TRUE
	Genetic Algorithm	20.611	21.189	0.285	0.466	0.956	TRUE

Table 11
Results of Welch's t-test (α = 0.01)

t-value	Critical Value (α = 0.01)	Hypothesis H₀ (t-test)
-11.795	2.736	FALSE

Conclusions

In this paper, an extension of the novel approach, the Triple Fuzzy Time Dependent Traveling Salesman Problem (3FTD TSP) model, was presented. The TD TSP has several benchmark instances in the literature (Eil51, Bier127, Eil76, Eil101, s250_1, and s250_2). Originally, in the TD TSP model, the costs between the nodes are time dependent, and this dependency relates to a predefined oblong-shaped area. In the extended 3FTD TSP model, the road conditions are represented by type-1 fuzzy sets. The uncertainty associated with the traffic jam areas and the fuzzy occurrence of the rush hours' phenomenon in time depend on several unknown or non-deterministic factors; thus, the uncertainty was represented by fuzzy sets of the respective universes. This way, determining an uncertain overall tour length (cost) was more efficient and practical.

Two meta-heuristic approaches, the Genetic Algorithm (GA) and the Discrete Bacterial Memetic Evolutionary Algorithm (DBMEA), were chosen to optimize the model due to their former success in similar (although non-fuzzy) problems. The simulation results confirm the efficiency (advantageous time and space complexity) and the predictability of the Discrete Bacterial Memetic Evolutionary Algorithm. Furthermore, it was found that the Discrete Bacterial Memetic Evolutionary Algorithm was able to converge in reasonable time for large instances with a maximum of 250 cities, without being stuck in a local optimum. By implementing a Genetic Algorithm-based approach as well, which was used for comparison, both being applied on the 3FTD TSP model, our extended model was able to adequately simulate complicated tour graphs and more complex actual cases.

Statistical tests were conducted on the bier127 benchmark, to validate the performance of the DBMEA, compared to the Genetic Algorithm. The results confirmed that the DBMEA significantly outperformed the GA in terms of solution quality at a 1% significance level. This further demonstrates the robustness of the DBMEA approach.

These findings prove the universality of the Triple Fuzzy Traveling Salesman Problem model, its' generality and applicability. Our future work will focus on applying, validating and comparing this novel model, on different transportation and various domains, to test the generality and efficiency.

References

[1] Applegate, D. L., Bixby, R. E., Chvátal, V., Cook, W. J.: The Traveling Salesman Problem: A Computational Study, Princeton University Press, Princeton, 2006, pp. 1-81

[2] Schneider, J.: The time-dependent traveling salesman problem, PhysicaA. 314, pp. 151-155, 2002

[3] Hurkała, J. Time-Dependent Traveling Salesman Problem with Multiple Time Windows. In Position Papers of the 2015 Federated Conference on Computer Science and Information Systems; Ganzha, M., Maciaszek, L., Paprzycki, M., Eds.; ACSIS: Marlton, NJ, USA, 2015; Volume 6, pp. 71-78

[4] Taş, D.; Gendreau, M.; Jabali, O.; Laporte, G. The Traveling Salesman Problem with Time-Dependent Service Times. Eur. J. Oper. Res. 2016, 248, 372-383

[5] Vu, D. M.; Hewitt, M.; Boland, N.; Savelsbergh, M. Dynamic Discretization Discovery for Solving the Time-Dependent Traveling Salesman Problem with Time Windows. Transp. Sci. 2019, 1-18, doi:10.1287/trsc.2019.0911

[6] Kóczy, L. T.; Földesi, P.; Tüű-Szabó, B. Enhanced discrete bacterial memetic evolutionary algorithm—An efficacious metaheuristic for the traveling salesman optimization. Inf. Sci. 2018, 460-461, 389-400

[7] Ban, H. B. An efficient two-phase metaheuristic algorithm for The Time Dependent Traveling Salesman Problem. RAIRO-Oper. Res. 2019, 53, 917-935

[8] Roy, A.; Manna, A.; Maity, S. A novel memetic genetic algorithm for solving traveling salesman problem based on multi-parent crossover technique, Decision Making. Appl. Manag. Eng. 2019, 2, 100-111

[9] Alberto V. Donati, Roberto Montemanni, Norman Casagrande, Andrea E. Rizzoli, Luca M. Gambardella,Time dependent vehicle routing problem with a multi ant colony system, European Journal of Operational Research, Volume 185, Issue 3, 2008, 1174-1191

[10] Zhang, T., Chaovalitwongse, W. A. & Zhang, Y. Integrated Ant Colony and Tabu Search approach for time dependent vehicle routing problems with simultaneous pickup and delivery. J Comb Optim 28, 288-309 (2014) https://doi.org/10.1007/s10878-014-9741-1

[11] Fox, K. R., Gavish, B., Graves, S. C., (1980) An n-constraint formulation of the (time-dependent) traveling salesman problem. Operations Research 28 (4), 1018-1021

[12] Malandraki, C., Dial, R. B., (1996) A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. European Journal of Operational Research 90, 45-55

[13] Fuzzy Based Solution to the Travelling Salesman Problem: A Case Study, Proceedings of the World Congress on Engineering and Computer Science 2017, Vol. II, WCECS 2017, October 25-27, 2017, San Francisco, USA

[14] Kóczy, L. T., Földesi, P., Tüű-Szabó, B., Ruba Almahasneh: Modeling of Fuzzy Rule-base Algorithm for the Time Dependent Traveling Salesman

Problem, Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), in publication, 2019

[15] Ruba Almahasneh, Tüű-Szabó, B., Földesi, Kóczy, L. T., Intuitionistic Fuzzy Model of Jam Regions and Rush Hours for the Time Dependent Traveling salesman Problem. Proceedings of the IFSA World Congress and NAFIPS Annual Conference, in publication, 2019

[16] Atanassov, RK.: Intuitionistic Fuzzy Sets, Fuzzy Sets and Systems 20, pp. 87-96, 1986

[17] Biswas, R.: On Fuzzy Sets and Intuitionistic Fuzzy Sets, Notes on Intuitionistic Fuzzy Sets 3, pp. 3-11, 1997

[18] Gau, W. L and Buehrer, D. J.: Vague Sets, IEEE Trans. Systems Man Cybernet, 23(2), pp. 610-614, 1993

[19] E. Szmidt, J. Kacprzyk, Intuitionistic Fuzzy Sets in Group Decision Making, NIFS 2 (1), pp. 11-14, 1996

[20] Ruba Almahasneh, Tüű-Szabó, B., Földesi, Kóczy, L.T., Extension of the Time Dependent Travelling Salesman Problem with Interval Valued Intuitionistic Fuzzy Model Applying Memetic Optimization Algorithm. ISMSI '20: Proceedings of the 2020 4th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, pp. 111-118, 2020

[21] Almahasneh R., Tuu-Szabo, Foldesi P., Koczy L. T. (2020) Fuzzy Set Based Models Comparative Study for the TD TSP with Rush Hours and Traffic Regions. In: Lesot MJ. et al. (eds) Information Processing and Management of Uncertainty in Knowledge-Based Systems. IPMU 2020. Communications in Computer and Information Science, vol. 1238. Springer, Cham.

[22] Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts -Towards Memetic Algorithms, Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, USA, 1989

[23] Nawa, N. E., Furuhashi, T.: Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, IEEE Tr. Fuzzy Systems 7, pp. 608-616, 1999

[24] Kóczy, T. L., Földesi, P., Tüű-Szabó, B.: Enhanced discrete bacterial memetic evolutionary algorithm- An efficacious metaheuristic for the traveling salesman optimization, Information Sciences, Volumes 460-461, pp. 389-400, 2018

[25] Zadeh, L. A: Fuzzy sets, Information and Control, 8 (3), pp. 338-353, 1965

[26] Mamdani, E. H.: Application of Fuzzy Algorithms for Control of Simple Dynamic Plant, IEEE Proc., Vol. 121, No. 12, pp. 1585-1588, 1974

[27] Voskoglou, M. Gr.: Comparison of the COG Defuzzification Technique and Its Variations to the GPA Index, American Journal of Computational and Applied Mathematics, 6(5), pp. 187-193, 2016

[28] Berenji H. R. (1992) Fuzzy Logic Controllers. In: Yager R. R., Zadeh L. A. (eds) An Introduction to Fuzzy Logic Applications in Intelligent Systems. The Springer International Series in Engineering and Computer Science (Knowledge Representation, Learning and Expert Systems), Vol. 165, Springer, Boston, MA.

[29] A. V. Patel, B. M. Mohan Fuzzy Sets and Systems,Volume 132, Issue 3, 16 December 2002, pp. 401-409

[30] Chakraverty S., Sahoo D. M., Mahato N. R. (2019) Defuzzification. In: Concepts of Soft Computing. Springer, Singapore

[31] Quoc Phan Tan, A Genetic Approach for Solving Minimum Routing Cost Spanning Tree Problem, IJMLC 2012 Vol. 2(4): 410-414 ISSN: 2010-3700

[32] Beasley, D., Bull, D. R. and Martin, R. R. (1993), "An overview of genetic algorithms: Part 2, research topics", University Computing, Vol. 15, pp. 170-181

[33] Földesi, P., Botzheim, J.: Modeling of loss aversion in solving fuzzy road transport travelling salesman problem using eugenic bacterial memetic algorithm, Memetic Computing, Volume 2, Issue 4, pp. 259-271, Springer-Verlag, 2010

[34] Tüű-Szabó, B., Ruba Almahasneh, Földesi, P., Kóczy, L. T.: A software tool for solving the Traveling Salesman Problem and related non-fuzzy and fuzzy optimization problems, Proceedings of the 14th European Symposium on Computational Intelligence and Mathematics (ESCIM 2022), October 2-5, 2023

[35] Balázs, K., Botzheim, J., Kóczy, T. L.: Comparison of Various Evolutionary and Memetic Algorithms, In: Integreted Uncertainty Management and Applications, AISC 68, pp. 431-442, Springer-Verlag Berlin Heidelberg, 2010