On the Design and Performance of a Novel Metaheuristic Solver for the Extended Colored Traveling Salesman Problem

Branko Miloradović[1], Eneko Osaba[2], Javier Del Ser[2,3], Vuk Vujović, and Alessandro V. Papadopoulos[1]

***Abstract*— Intelligent transportation systems face various challenges, including traffic congestion, environmental pollution, and inefficient transportation management. Optimizing routes and schedules for efficient delivery of goods and services can mitigate the aforementioned problems. Many transportation and routing problems can be modeled as variants of the Traveling Salesmen Problem (TSP) depending on the specific requirements of the scenario at hand. This means that to efficiently solve the routing problem, all locations have to be visited by the available salesmen in a way that minimizes the overall makespan. This becomes a non-trivial problem when the number of salesmen and locations to be visited increases. The problem at hand is modeled as a special TSP variant, called Extended Colored TSP (ECTSP). It has additional constraints when compared to the classical TSP, which further complicates the search for a feasible solution. This work proposes a new metaheuristic approach to efficiently solve the ECTSP. We compare the proposed approach to existing solutions over a series of test instances. The results show a superior performance of our metaheuristic approach with respect to the state of the art, both in terms of solution quality and algorithm's runtime.**

I. Introduction

Among all the optimization problems that can be formulated in scenarios related to Transportation and Mobility, the Travelling Salesman Problem (TSP) can be regarded as the most widely studied one in the literature for decades [1]. In essence, the TSP seeks to discover the shortest route that a salesman can take to visit a set of cities and return to its starting point, without visiting any city more than once. This classic optimization problem is representative of a myriad of practical use cases in transportation and logistics, including the optimization of delivery routes, travel itinerary planning, flight scheduling, or the design of supply chain networks, to mention a few [2]. Furthermore, finding an optimal solution to the TSP can help minimize a diversity of objectives related to the optimized routes, such as transportation costs, delivery times or fuel consumption.

Since its inception in the early XIX century, the combinatorial complexity of the TSP problem and its variants has spurred the search for approximated and heuristic solvers to find near-optimal solutions to these problems, particularly for large-scale instances with a high number of cities [3], [4]. Optimization algorithms as such inspired by principles and behaviors observed in nature and physical phenomena (namely, bio-inspired optimization) have become prevalent over the years for this purpose. The adoption of this family of solvers has risen sharply within the research community when the complexity of the scenario being modeled requires adding further ingredients to the seminal TSP formulation, including the presence of multiple salesmen, time windows, or capacity constraints of the vehicles. Consequently, well-established flavors of the TSP problem have been studied over the years, such as the Multiple Traveling Salesman Problem (MTSP, [2]), which aims to find the optimal set of routes for several salesmen such that each city is visited exactly once and the total distance traveled by all salesmen is minimized. Likewise, the Time-Dependent Traveling Salesman Problem (TD-TSP, [5]) targets the scenario in which the travel time between cities vary over time due to weather conditions, traffic status, and other impacting factors. Similarly, the Dynamic Traveling Salesman Problem (DTSP, [6]) models the real-world situation in which the set of available cities vary over time. Several surveys have exposed the vigor of this area and the multiplicity of problem formulations rooted on the seminal TSP, reviewing the diverse optimization techniques used to address them efficiently [7].

In this context, this work focuses precisely on one of these TSP variants proposed in the recent past: the so-called Extended Colored TSP (ECTSP [8]). ECTSP has practical applications in the Multi-Agent Mission Planning [9] domain. This problem considers multiple salesmen, each qualified to access certain cities (represented by colors), with precedence constraints between cities, several possible source and destination locations, and heterogeneous characteristics of the salesmen (i.e., different velocities, starting points, qualifications to visit the cities, etc). Therefore, the time for a salesman to visit all cities involved in a route depends on its qualification and speed. Precedence constraints impose an ordering constraint between cities, such that some cities must be visited before or after some other cities along a given route. The work in [8] also defined genetic search operators and a precedence constraint reparation method to deal with this problem formulation effectively. Later, the follow-up work in [10] extended this prior work by reformulating the ECTSP as a Mixed-Integer Linear Programming problem (MILP), changing the way precedence constraints can be formulated. CPLEX was used to solve this alternative formulation, showing that it can be used for small instances of this problem. Recently, a

This work was supported by the Swedish Research Council (VR) with the PSI project (No. #2020-05094), by the Knowledge Foundation (KKS) with the FIESTA project (No. #20190034), and the European Union's H-2020 research and innovation programme under grant agreement No 101004590 (URBANAGE).

[1]Department of Intelligent Future Technologies, Mälardalen University, Sweden. E-mail: branko.miloradovic@mdu.se

[2]TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Spain. E-mail: eneko.osaba@tecnalia.com

[3]University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain.

similar problem to ECTSP has been formulated, denoted as Precedence CTSP (PCTSP) [11].

This paper builds upon these previous efforts to propose a new metaheuristic solver for the ECTSP solver which incorporates novel algorithmic components in its search procedure. More concretely, the main contributions of this method are *i*) a new solution creation method; and *ii*) modified 2-opt and 3-opt operators [12] that account for the needed satisfaction of the precedence constraints imposed in the problem formulation. Experimental results over a set of public ECTSP instances are discussed and compared to those produced by the previously proposed solvers for this problem. As evinced by our experimental results, the proposed algorithm outperforms existing approaches in the discovered routes for the salesmen, improving with statistical significance its predecessors.

The rest of the manuscript is structured as follows: first, Section II briefly reviews the state of the art related to TSP, TSP variants, and meta-heuristic algorithms used for these problems, towards clearly stating the contribution of this work. Next, Section III describes the problem formulation of ECTSP, whereas Section IV shows in detail the proposed algorithm, including the creation, evaluation, and mutation of the population of candidate solutions held during the search. Section V presents the experimental setup and discusses on the results obtained therefrom. Finally, Section VI ends the paper with concluding remarks and an outlook toward future research lines departing from our findings.

II. Related Work

Before delving into the details of the ECTSP and the proposed algorithm, we briefly revisit the long history between TSP variants and metaheuristic optimization algorithms (Section II-A), followed by an examination of formulations that comprise multiple salesmen, different qualifications (colors) and precedence constraints (Section II-B). Finally, we frame the novelty and contribution of this work within the literature reviewed in this section (Section II-C).

A. TSP variants and metaheuristic optimization

The TSP has grown to be one of the most studied problems in operations research and computational intelligence. Over the years, the TSP has demonstrated a great flexibility to model real-world optimization problems in fields such as robotics [13], agriculture [14], or disaster logistics [15].

Several intelligent approaches have been adopted for efficiently solving TSP and its variants, even in recent years. Among these approaches, classical techniques such as genetic algorithms [16], or simulated annealing [17] can be found. More recent methods have also been employed for this purpose, such as Ant Colony Optimization [18] or Particle Swarm Optimization [19]. Besides these widely known algorithms, the TSP has been used as a benchmark problem for assessing the performance of modern biologically inspired metaheuristics [20], [21], such as the Bat Algorithm [22], or the Firefly Algorithm [23]. Recently, the TSP has also been tackled by revolutionary computing paradigms, such as quantum computing [24].

All this vigorous scientific activity around TSP confirms that the interest behind this problem and its variations remains alive today. For readers interested in this problem, we refer to the comprehensive surveys in [1], [7].

B. Precedence constraints, coloring, and multiple salesmen

Apart from its canonical formulation, many different TSP variants have been addressed over the years, aiming to model specific features present in real-world logistics and transportation problems. We now list known variants of the TSP that connect to the ECTSP problem tackled herein:

- TSP with time windows: This is arguably the most popular variant of the TSP, in which each node has an associated time window. Thus, each node must be visited at a time that falls within its window [25].
- Generalized TSP: In this variant, the group of nodes is organized in groups or *clusters*. The main objective is to find a route that visits each cluster exactly once (i.e., visit a single node per cluster) [26].
- Open TSP: In this formulation, the goal is to find the shortest route by starting from a predefined node, and without needing to complete a cycle (i.e., the salesman does not need to return to the starting point of the route) [27].
- Asymmetric TSP: This variant is characterized by having asymmetric costs, meaning that going from one node to another has a different cost than the reverse trip [28].

Apart from these variants, the following TSP problem formulations are especially interesting for our study:

- Multiple TSP: As already mentioned in the introduction, in this problem a fixed group of vehicles is available, which must be used for visiting all the available nodes. Also, each route should start and finish in a defined starting node [2].
- Colored TSP: In this formulation, nodes are divided into n exclusive groups and one shared set. The goal is to find n shortest routes, considering that an exclusive set must be visited in the same route, while shared cities can be visited by any route [29].
- TSP with precedence constraints: some nodes present precedence restrictions. In other words, prior to visiting a node, a list of preceding nodes must be visited before [30].
- TSP with pickup and delivery times and/or capacity: In this variant, the traveling salesman has an associated capacity, whereas all nodes have a predefined demand/capacity that may reflect the pickup or drop-off of materials/goods. The objective is to find the shortest route, not exceeding the capacity of the traveler [31].

C. Contribution

In light of the reviewed literature, the contribution of this paper is the improvement of previous attempts at efficiently solving the ECTSP problem, showing that the exploitation of problem-specific knowledge in the design of the search algorithm can improve the quality of the solutions and reduce its execution time.

III. Problem Description

In this section, we briefly describe the problem addressed in this work (ECTSP). Detailed mathematical formulations are given by Miloradović *et al.* [8], [10].

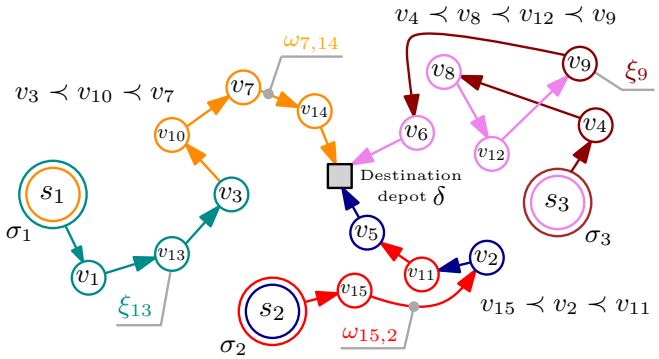

Fig. 1: Diagram describing a solution to an ECTSP problem with $M = 3$ salesmen, $N = 15$ cities, $K = 6$ colors, and a single destination depot ($|\Delta| = 1$).

Let $s \in \mathcal{S}$ be a salesman, in a set $\mathcal{S} := \{s_1, s_2, \ldots, s_M\}$ of M salesmen, that need to visit N cities in a set $\mathcal{V} := \{v_1, v_2, \ldots, v_N\}$. Also, let c be a color that varies in a set $\mathcal{C} := \{c_1, c_2, \ldots, c_K\}$ of K colors. Each salesman s_m must start its tour from a source depot $\sigma_m \in \Sigma$ and finish at a destination depot $\delta_m \in \Delta$. Source and destination depots are not considered to be cities. Traversing from either source depots or nodes to other nodes or destination depots is associated with a cost of ω_{ij}. In addition, every node has a weight ξ_n that is added to the total cost when the node is visited. This weight represents the duration salesman s_m is required to stay in city v_n. Every city is associated with one color from set $\mathcal{C}$, and every salesman is associated with a subset of colors from set $\mathcal{S}$. If the intersection between the color of the city and the subset of colors of a salesman is not an empty set, the given salesman can visit that city. In addition, cities may have precedence relations, i.e., it might be needed for city v_n to be visited before city $v_{n'}$. This is defined with $\pi_{n,n'} = 1 \iff n \prec n'$, and 0 otherwise.

This problem can be formulated over a directed graph $\mathcal{G} = (\widetilde{\mathcal{V}}, \mathcal{E})$, with $\widetilde{\mathcal{V}}$ being a set of vertices, and $\mathcal{E} : \widetilde{\mathcal{V}} \times \widetilde{\mathcal{V}} \to \mathbb{R}_0^+$ being a set of edges. The solution to the aforementioned problem is a set of Hamiltonian paths, one for each salesman. Note that a Hamiltonian path can contain 0 vertices. A Hamiltonian path is defined as a path in a graph that visits each vertex in a set of vertices exactly once. Finally, we can define the goal as the allocation of cities to salesman in a way to minimize the overall cost (i.e., the sum of all edge and node weights along the routes of the salesmen), while respecting color and precedence constraints.

A visual example of a possible solution to an ECTSP problem is given in Fig. 1. The problem consists of three salesmen ($S1$, $S2$, and $S3$), 14 cities in total allocated to the available salesmen. The cities have 6 different colors in total. In the example, there is only one destination depot.

IV. Proposed Solver

The proposed algorithm, hereafter referred to as MDUTEC, follows the well-established paradigm of evolutionary algorithms. The initial population is seeded with solutions drawn from the feasible search space, and progressively improved through a number of search operators. More specifically, the algorithm begins by creating an initial population (Section IV-B) in the feasible region of the search space. In order to ensure that the candidate solutions are feasible, each solution undergoes a precedence constraint reparation procedure, described in Section IV-C. The new candidate solutions are introduced in the population through variation operators explained in Section IV-E. Finally, the optimization process stops when a predefined time limit is reached. In the rest of this section, we elaborate on each part of the algorithm.

A. Solution encoding

A candidate solution is encoded as an array of length $N \cdot M$, where N represents the total number of cities and M is the total number of agents[1] available for a specific mission. A graphical representation of a candidate solution is given in Fig. 2. The elements in the array are integers ranging from 1 to N depending on the number of cities assigned to a specific agent. These integers also determine the order of cities to be done in ascending order. If we focus on the provided example in Fig. 2, city 1 (C1) has ID 1, which means it will be done first by agent 1, C3 has ID 2, which means it is the second city to be performed, and finally, C2 has ID 3, which is consequently done as the 3rd city in the plan. Agent 2 has only one city, i.e., C4 with ID 1.

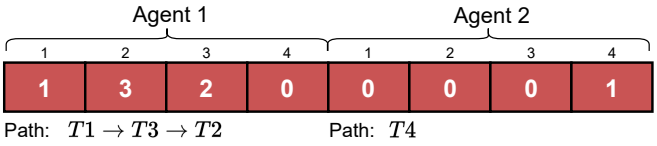

Fig. 2: An example of an encoded solution for a plan comprising $M = 2$ agents and $N = 4$ cities.

B. Creating the initial population

The initial population is created based on a nearest-neighbor heuristic. Each candidate solution is formed in the same manner. First, the list of all cities to be done is created and shuffled. The list is then iterated through, and each city is allocated to the closest agent (as per the coordinates of every node and the agents) or the closest previously allocated city. This process takes into account color constraints, but not precedence constraints. In order to ensure that initial candidate solutions fall within the feasible region of the search space, a precedence constraint reparation algorithm is developed.

[1]We use the terms *salesman* and *agent* interchangeably in this work.

C. Reparation of precedence constraints

Each candidate solution is submitted to a Precedence Violation Check (PVC). It is important to note that in this context, precedence strictly refers to the ordering of cities. For the sake of simplicity and without loss of generality, we assume only precedence relations between two cities, however, the transitive property still applies, i.e., if $T1 \prec T2$ and $T2 \prec T3$, then $T1 \prec T3$. The PVC determines whether there is an ordering violation between the cities, and identifies the type of conflict. In general, there can be two types of conflict. The first one occurs when both cities are allocated to the same agent but in reverse order. In the second case, one of the cities is allocated to a different agent, and by the definition of the problem, this is not allowed. When conflicts are identified, the algorithm proceeds with solution reparation. In the first case, the reparation approach is straightforward: the location of the cities in the plan is swapped. This simple move is enough to solve the precedence violation. The second case is more complicated as first, the agents to which both cities are allocated have to be found. Then a *coin flip* is performed to determine which city should be moved. This procedure is repeated until there are no more precedence violations in the candidate solutions and, consequently, in the population.

D. Population evaluation

Candidate solutions are evaluated based on the objective function in use: the sum of makespans over all salesmen. After all candidate solutions have been evaluated and their cost calculated, the population is sorted based on the cost. In addition, a Hamming distance (H) is calculated between the best candidate solution in the population and all other candidate solutions. This value is used to determine if the algorithm should take a bigger or a smaller move when creating new candidate solutions.

E. Creation of new candidate solutions

The process of creating new candidate solutions is provided as a pseudocode in Algorithm 1. In the first step, we find the best solution in the population. In the next step, the Hamming distance (H) between the candidate solution and the best solution in the population is calculated. Following is a random selection of value rnd in the range from 1 to H. After this the variation operations are performed, namely *Insert city*, *Insert to Closest (ITC)*, and *Swap city*, in that order. For every candidate solution, this process is repeated r times, where r is a tunable parameter. In our experiments presented in Section V, we used the value $r = 3$, as this value showed to give a good balance between computational complexity and solution quality. Finally, the value of rnd is used to determine if the algorithm should perform 2-opt or 3-opt heuristics. More specifically, if the rnd value is larger than a quarter of the number of total cities (N), a bigger leap in the search space is taken by using 3-opt heuristics. Otherwise, 2-opt is used. These steps are repeated for all candidate solutions in the population.

a) Insert city: The insert city operator starts by randomly selecting an agent ($A1$) and randomly selecting a city ($C1$) from the set of cities allocated to agent $A1$. City $C1$ is then removed from its location in the plan and inserted in the randomly determined new location in the plan. It can happen that city $C1$ is allocated to the same agent $A1$ from where it has been previously removed, just in a different location in the plan, or it can be inserted in the plan of some other available agent in the mission. In the case city $C1$ has a precedence relation with some other city, the insertion is limited to only $A1$ with respect to precedence constraints. For example, if $C1$ is the 5th city in the plan of agent $A1$, and it is a predecessor city to the 7th city in agent's $A1$ plan, then city $C1$ can only be inserted between location 1 and 7 in agent's $A1$ plan.

Algorithm 1 Creation of new candidate solutions

```
function CREATENEWSOLUTION(candidateSol)
    bestSol ← GETBESTSOLUTION(population)
    H ← CALCHAMMINGDIST(candidateSol, bestSol)
    rnd ← RANDOMINTEGER(1, H)
    for 1 : r do
        INSERTCITY(candidateSol)
        INSERTTOCLOSEST(candidateSol)
        SWAPCITY(candidateSol)
    if rnd < N/4 then
        TWOOPT(candidateSol)
    else
        THREEOPT(candidateSol)
    return candidateSol
```

b) Swap city: The swap city operator starts by randomly selecting two agents ($A1$ and $A2$). It may also happen that the two selected agents are the same. In the next step, we check if color sets of agents $A1$ and $A2$ have intersecting elements. If that is the case, the next move is to find all cities whose color is the element of the set of intersecting colors of agents $A1$ and $A2$. Randomly, two cities should be selected for swapping, unless they are having precedence relationship between them. In case they have a precedence relationship with other cities, those cities should be moved as well. For example, if $C1$ from $A1$ is selected for swapping with $C2$ allocated to $A2$, and $C2$ has precedence relation with city $C3$ on $A2$, then both cities $C2$ and $C3$ will be moved to $A1$, and $C1$ will be moved to $A2$.

c) Insert To Closest: This operator starts by selecting a random agent ($A1$) from the set of available agents. The next step is to randomly select a city ($C1$) among the cities allocated to agent $A1$. In the next step, we create a list of the closest cities to city $C1$ and select a city ($C2$) from the list. Cities that are closer to $C1$ have a higher probability of being chosen. At this point, another check is performed to determine if it is better (shorter path) to add $C1$ before or after $C2$. When the insert position is determined, $C1$ is removed and inserted at the new position in the plan. In case $C1$ had precedence relations, there are additional steps

to be done. The first one is to determine if $C1$ is succeeding or preceding the city. Depending on this, a list of cities is created, either from the start of the plan up to the location of $C1$ in the plan if $C1$ is a preceding city, or from the location of $C1$ in the plan until the end of the plan if $C1$ is a succeeding city. The city that has precedence relations with $C1$ is then inserted in the appropriate sequence at the location in the path that is closest to it. In order to better illustrate how ITC operator works, we provide Fig. 3. The figure shows two agents (salesmen) and their routes (Fig. 3 up) and their routes after applying ITC operator (Fig. 3 down). ITC is applied to city v_3 which is added to the route of salesman s_2. It is added in the route of s_2 after the closest city to it, which is city v_8.

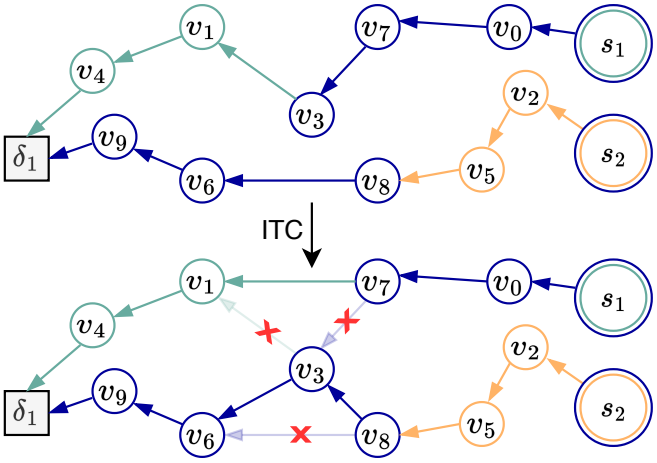

Fig. 3: An example of an ITC operation.

d) 2-opt: The 2-opt algorithm was first proposed by Croes [32] to solve the TSP problem. In order to be used for the ECTSP tackled in this work, this operator must be adapted. In our algorithm, 2-opt is used to perform a local refinement of the plan, i.e., it is applied to each agent's plan. The algorithm loops over cities and in each iteration, two points (cities) are selected ($C1$ and $C2$), where $C2$ always comes after $C1$ in the plan. After two cities have been selected, the plan is refined in the way that the sequence starting from the first city in the plan up to $C1$ is unchanged. The sequence of cities from $C1$ to $C2$ is reversed, and finally, the sequence from $C2$ until the end of the plan is unchanged. What has been so far described is the typical 2-opt implementation, the rest of the paragraph described the added parts to the 2-opt heuristics. If the new plan has a lower overall cost, the algorithm moves to the next step, which is precedence constraint violation checking. If reversing the sequence of cities from $C1$ to $C2$ did not violate precedence constraints, the new plan is inserted in the solution. Otherwise, another step is performed that checks if adding $C2$ immediately after $C1$ in the plan would improve the solution cost. If this step does not yield positive results, the new plan is discarded, new $C1$ and $C2$ are selected, and the process is repeated until all city combinations are tried out.

e) 3-opt: The 3-opt algorithm is an improved version of the 2-opt algorithm at a higher computational cost. The algorithm loops over cities and in each iteration, 3 cities are selected ($C1$, $C2$, and $C3$), where $C1$ comes before $C2$ which comes before $C3$ in the plan. We remove the three routes from the plan that connect $C1 \rightarrow C1+1$, $C2 \rightarrow C2+1$, and $C3 \rightarrow C3+1$. Now, there are 7 possible new ways of reconnecting the cities. In total, 3 of those 7 ways are equivalent to a single 2-opt move. The other 4 cannot be achieved with a single 2-opt move, and we focus on them. We calculate the impact of each move on the overall cost and sort the moves based on their impact. In the next step, each of the moves is tested to assess whether it satisfies all precedence constraints. If so, the move is applied. Otherwise, the move is discarded, and the algorithm tests the next move in the list. Depending on the selected move, different parts of the path are reversed to create a new plan. If no move is applicable, the algorithm continues iterating through the city list until all city combinations are considered.

V. Experiments and Results

In order to assess the performance of the proposed MDUTEC algorithm, several experiments have been performed to compare its computational efficiency and the quality of its discovered routes when compared to the best-known solver for this problem, namely, the so-called Genetic Mission Planner (GMP) from [10]. Both solvers are implemented in C++ programming language. The comparison is performed over a set of 10 ECTSP instances with increasing complexity. Table I summarizes the configuration of such benchmark instances. Each (algorithm, instance) combination is run for 30 times with different seeds, in order to account for the statistical variability of the fitness of the solutions and to reduce the impact of reaching a good solution by simple chance on the conclusions drawn from the results. The evaluation is done on an x64 platform with i9-9980XE @3.8GHz CPU and 128GB of DDR4 memory. The time limit has been set for both algorithms to 300 seconds and for comparison purposes, both algorithms are running on a single thread. Instances used in this paper have been released in a public repository (https://github.com/

TABLE I: Parameters of the ECTSP benchmark instances, with *N* being the number of cities, *M* the number of salesmen, $|\Delta|$, and #PC is the number of precedence constraints in the mission.

ID	N	M	$\|\Delta\|$	K	#PC
1	10	1	1	1	1
2	30	2	1	2	5
3	50	3	2	2	5
4	75	4	2	3	13
5	100	5	3	3	6
6	150	6	3	3	25
7	200	7	4	3	14
8	300	8	4	3	51
9	400	9	5	3	60
10	500	10	5	3	30

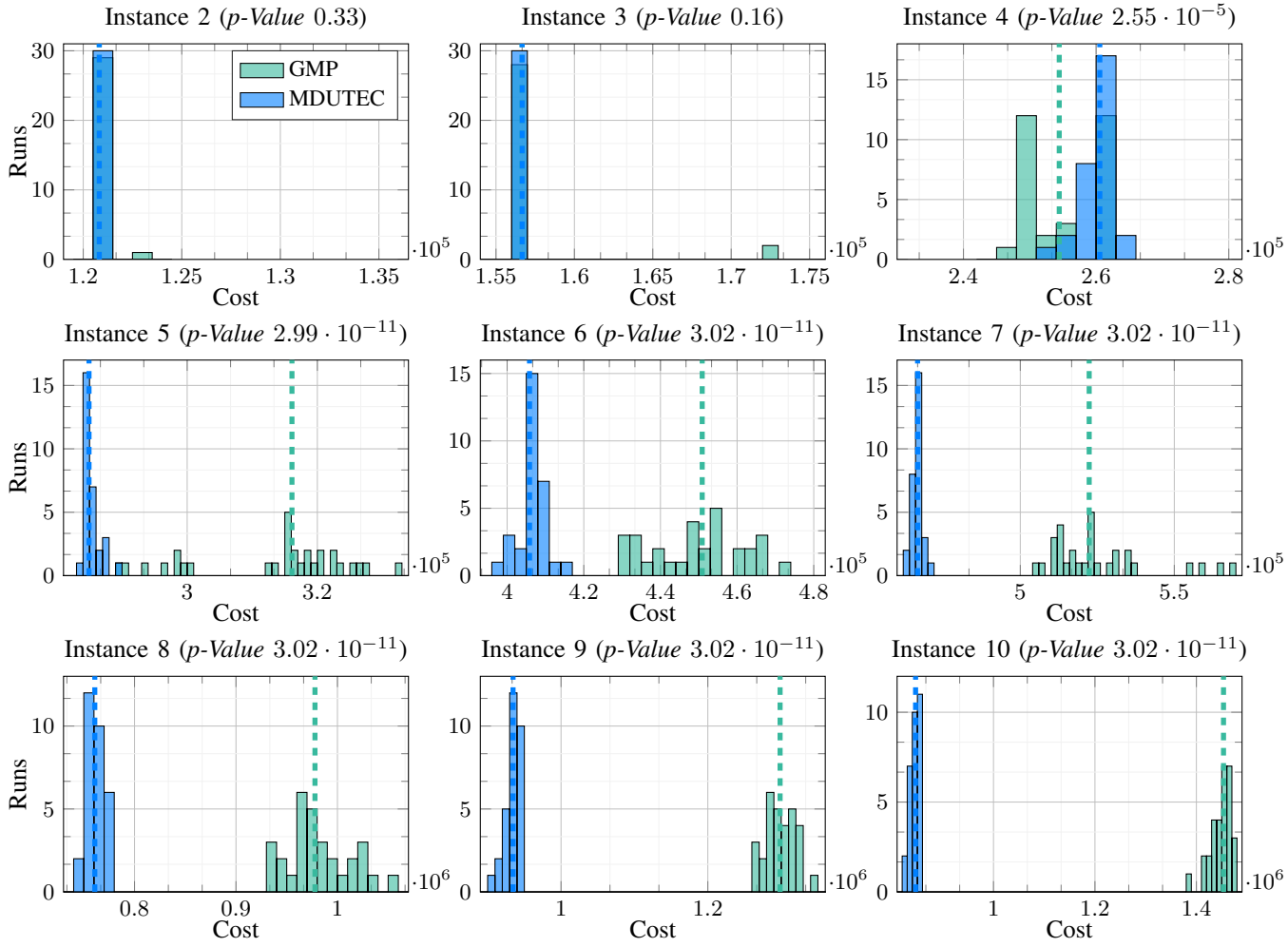

Fig. 4: Histograms of the best routes found by MDUTEC and GMP over the 30 runs that each solver is run for instances 2 to 10. The X-axis shows the cost of the routes (fitness), whereas the Y-axis is the number of runs ending with a solution that has such cost. Vertical dashed lines in the figures represent the median value of each histogram. We note that in some cases, bins overlap, e.g., in Instance 2, 29/30 runs have the same final cost.

mdh-planner/ECTSP) for the sake of reproducibility and to support follow-up studies.

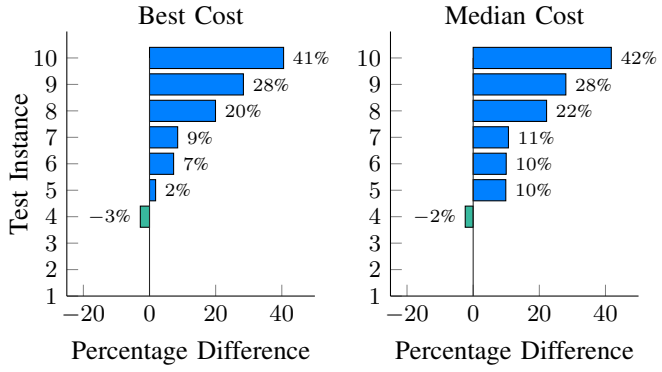

Fig. 5: Comparison between GMP and MDUTEC in terms of the best solution quality (left) and median cost (right).

A. Results and discussion

To begin with, Fig. 4 shows the histogram of the overall cost of the routes (the fitness of the best solutions) discovered by the GMP and MDUTEC solvers over each ECTSP instance. We note that Instance 1 is excluded from the analysis of the results, since both algorithms were able to find the same solution in each run. It can be observed that both solvers perform similarly in terms of median solution quality for Instances 2 and 3, with GMP having one and two solution outliers, respectively. The similarity between the results can be seen in Fig. 5, where for Instances 2 and 3, there is no difference in terms of the best solution found or median cost.

However, there can be observed a different pattern in Fig. 6, which depicts the empirical cumulative density function (ECDF) of both algorithms over time. In general, it can be observed that it takes longer for GMP to reach 10% of the best-known solution, by the order of magnitude. Instance 4 is the only instance where GMP outperforms MDUTEC by 3% on the best solution (Fig. 5 on the left) and by 2% on

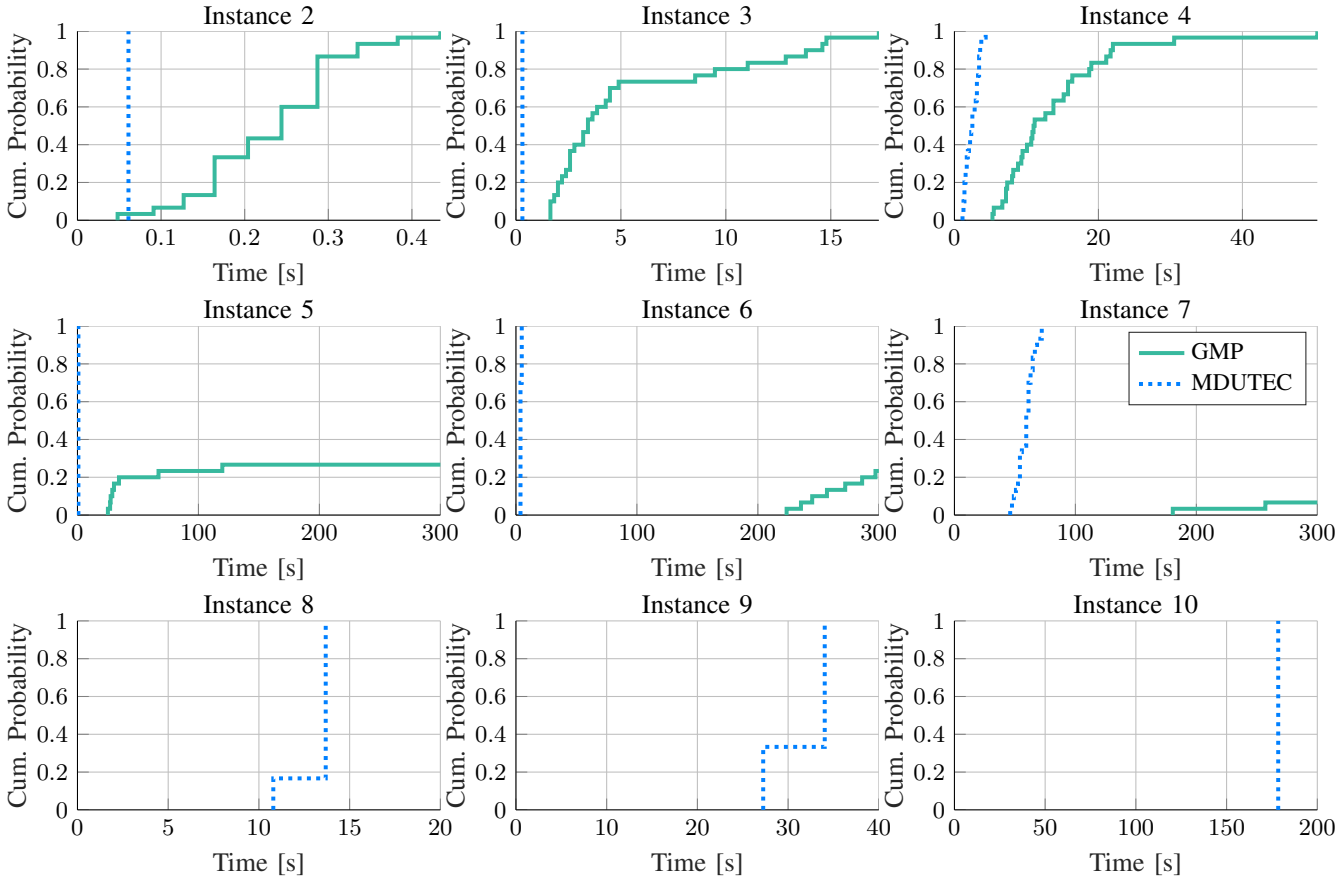

Fig. 6: Empirical CDF for Instances 2–10. The target value is within 10% of the best known solution for each instance.

median cost (Fig. 5 on the right). However, Fig. 6 shows that GMP takes a much longer time to reach the solution quality of within 10% of the best-known one. This means that even in the case when the GMP outperforms the MDUTEC solver in terms of solution quality, it requires a much longer time to converge. Starting from Instance 5 until Instance 10, the gap between the two solvers widens progressively, as MDUTEC outperforms GMP in each of such instances. In Fig. 5 it is shown that the difference in solution quality increases as the complexity of the test instances increases, and it ranges from 2% for Instance 5 to 41% for Instance 10. The median cost difference follows a similar trend and the difference is even larger, ranging from 10% in Instance 5 to 42% in Instance 10. In the case of Instances 8 to 10, the GMP solver is unable to find any solution that is within 10% of the best-known solution (Fig. 6).

Fig. 4 also shows p-values computed by a non-parametric Mann-Whitney-Wilcoxon test between the sets of results obtained by GMP and MDUTEC. This test is used to gauge the statistical significance of the observed performance gaps, since the distribution of the fitness values over which the comparison is made can be highly skewed and long-tailed. Should this occur, discussing on average performance statistics may give rise to misleading conclusions. To circumvent this potential issue, the median value was used instead of the mean. The p-value test shows that the gap by which MDUTEC outperforms GMP in Instances 5–10 is statistically significant. Conversely, MDUTEC is surpassed by GMP in Instance 4 with a p-value of $2.55 \cdot 10^{-5}$, which even if significant under typical confidence levels (e.g. $\alpha = 0.05$), is notably larger than p-values obtained for larger instances.

Based on the above analysis, it is fair to conclude that for MDUTEC performs clearly better than GMP, with differences that progressively increase with the complexity of the ECTSP problem instance.

VI. Conclusions and Future Research

This paper has elaborated on a variant of the TSP routing problem known as ECTSP, which extends the seminal TSP formulation by including additional ingredients such as multiple salesmen, color, and precedence constraints. To efficiently solve this problem, we have proposed a new metaheuristic solver (MDUTEC) that incorporates multiple heuristics extended and adapted to this kind of problem, such as 2-opt, 3-opt, and the novel Insert to Closest (ITC) operator. To comparatively evaluate the performance of the proposed algorithm, a discussion has been held on the performance of MDUTEC and the best solver known so far for this problem (GMP) when solving a set of 10 test instances. These two solvers have been compared in terms of solution quality and convergence. The experimental results have revealed that

MDUTEC outperforms GMP in most test instances. In the case of solution quality, MDUTEC is able to perform equal to or better than its counterpart in 9 out of 10 tests. In what refers to convergence, MDUTEC was found to converge faster than GMP in all 10 test instances.

The promising results presented in this work suggest several research directions to be pursued in the near future. Among them, we plan to investigate other variation operators commonly used for TSP. Our idea is to study how to adapt them to the specifics of the ECTSP variant, and to verify whether such adaptations reflect on a performance improvement of the metaheuristic algorithm presented in this work. We will also address more complex formulations of this problem. For instance, the dynamic ECTSP, in which costs associated to nodes and links in the underlying graph vary over time; or the multi-objective ECTSP, in which several conflicting cost functions are defined on nodes and edges, requiring the search for Pareto-optimal routes.

REFERENCES

[1] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: an overview of applications, formulations, and solution approaches," *Traveling salesman problem, theory and applications*, vol. 1, 2010.

[2] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy," *Computer Science Review*, vol. 40, p. 100369, 2021.

[3] C. Jiang, Z. Wan, and Z. Peng, "A new efficient hybrid algorithm for large scale multiple traveling salesman problems," *Expert Systems with Applications*, vol. 139, p. 112867, 2020.

[4] X. Dong and Y. Cai, "A novel genetic algorithm for large scale colored balanced traveling salesman problem," *Future Generation Computer Systems*, vol. 95, pp. 727–742, 2019.

[5] A. Lucena, "Time-dependent traveling salesman problem–the deliveryman case," *Networks*, vol. 20, no. 6, pp. 753–763, 1990.

[6] T. Cheong and C. C. White, "Dynamic traveling salesman problem: Value of real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 619–630, 2011.

[7] E. Osaba, X.-S. Yang, and J. Del Ser, "Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics," *Nature-Inspired Computation and Swarm Intelligence*, pp. 135–164, 2020.

[8] B. Miloradović, B. Çürüklü, M. Ekström, and A. V. Papadopoulos, "A genetic algorithm approach to multi-agent mission planning problems," in *Operations Research and Enterprise Systems*. Springer Int. Publishing, 2020, pp. 109–134.

[9] B. Miloradović, "Multi-Agent Mission Planning," Ph.D. dissertation, Mälardalen University, 2022.

[10] B. Miloradović, B. Çürüklü, M. Ekström, and A. V. Papadopoulos, "GMP: A genetic mission planner for heterogeneous multirobot system applications," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 627–10 638, 2021.

[11] X. Xu, J. Li, M. Zhou, and X. Yu, "Precedence-constrained colored traveling salesman problem: An augmented variable neighborhood search approach," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9797–9808, 2021.

[12] L. Muyldermans, P. Beullens, D. Cattrysse, and D. Van Oudheusden, "Exploring variants of 2-opt and 3-opt for the general routing problem," *Operations research*, vol. 53, no. 6, pp. 982–995, 2005.

[13] S. Trigui, O. Cheikhrouhou, A. Koubaa, U. Baroudi, and H. Youssef, "FL-MTSP: a fuzzy logic approach to solve the multi-objective multiple traveling salesman problem for multi-robot systems," *Soft computing*, vol. 21, pp. 7351–7362, 2017.

[14] X. Li, Z. Ma, X. Chu, and Y. Liu, "A cloud-assisted region monitoring strategy of mobile robot in smart greenhouse," *Mobile Information Systems*, vol. 2019, pp. 1–10, 2019.

[15] O. Cheikhrouhou, A. Koubâa, and A. Zarrad, "A cloud based disaster management system," *Journal of Sensor and Actuator Networks*, vol. 9, no. 1, p. 6, 2020.

[16] T. George and T. Amudha, "Genetic algorithm based multi-objective optimization framework to solve traveling salesman problem," in *International Conference on Advancements in Computing and Management (ICACM 2019)*. Springer, 2020, pp. 141–151.

[17] İ. İlhan and G. Gökmen, "A list-based simulated annealing algorithm with crossover operator for the traveling salesman problem," *Neural Computing and Applications*, pp. 1–26, 2022.

[18] Y. Wang and Z. Han, "Ant colony optimization for traveling salesman problem based on parameters optimization," *Applied Soft Computing*, vol. 107, p. 107439, 2021.

[19] Y. Cui, J. Zhong, F. Yang, S. Li, and P. Li, "Multi-subdomain grouping-based particle swarm optimization for the traveling salesman problem," *IEEE Access*, vol. 8, pp. 227 497–227 510, 2020.

[20] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. C. Coello, and F. Herrera, "Bio-inspired computation: Where we stand and what's next," *Swarm and Evolutionary Computation*, vol. 48, pp. 220–250, 2019.

[21] R.-E. Precup, E.-I. Voisan, R.-C. David, E.-L. Hedrea, E. M. Petriu, R.-C. Roman, and A.-I. Szedlak-Stinean, "Nature-inspired optimization algorithms for path planning and fuzzy tracking control of mobile robots," *Applied Optimization and Swarm Intelligence*, pp. 129–148, 2021.

[22] Y. Saji and M. E. Riffi, "A novel discrete bat algorithm for solving the travelling salesman problem," *Neural Computing and Applications*, vol. 27, pp. 1853–1866, 2016.

[23] L. Zhou, L. Ding, X. Qiang, and Y. Luo, "An improved discrete firefly algorithm for the traveling salesman problem," *Journal of Computational and Theoretical Nanoscience*, vol. 12, no. 7, pp. 1184–1189, 2015.

[24] E. Osaba, E. Villar-Rodriguez, and I. Oregi, "A systematic literature review of quantum computing for routing problems," *IEEE Access*, vol. 10, pp. 55 805–55 817, 2022.

[25] H. Kona, A. Burde, and D. Zanwar, "A review of traveling salesman problem with time window constraint," *IJIRST-International Journal for Innovative Research in Science & Technology*, vol. 2, 2015.

[26] I. Khan, M. K. Maiti, and K. Basuli, "Multi-objective generalized traveling salesman problem: a decomposition approach," *Applied Intelligence*, vol. 52, no. 10, pp. 11 755–11 783, 2022.

[27] A. Kirimtat, O. Krejcar, M. F. Tasgetiren, and E. Herrera-Viedma, "Multi-performance based computational model for the cuboid open traveling salesman problem in a smart floating city," *Building and Environment*, vol. 196, p. 107721, 2021.

[28] T. Öncan, İ. K. Altınel, and G. Laporte, "A comparative analysis of several asymmetric traveling salesman problem formulations," *Computers & Operations Research*, vol. 36, no. 3, pp. 637–654, 2009.

[29] P. He, J.-K. Hao, and Q. Wu, "Grouping memetic search for the colored traveling salesmen problem," *Information Sciences*, vol. 570, pp. 689–707, 2021.

[30] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *European Journal of Operational Research*, vol. 140, no. 3, pp. 606–617, 2002.

[31] H. Hernández-Pérez and J.-J. Salazar-González, "A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery," *Discrete Applied Mathematics*, vol. 145, no. 1, pp. 126–139, 2004.

[32] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958.