Research Article

Maximizing Total Net Profit for Traveling Salesman Problem with Profits Using Metaheuristic Algorithms

Eyüp Ensar Işık[1*], Mısra Şimşir[2*]

1 Department of Industrial Engineering, Yildiz Technical University, 34349 Istanbul, Turkey, Orcid ID: 0000-0002-9180-0243, eeisik@yildiz.edu.tr,
2 Department of Industrial Engineering, Yildiz Technical University, 34349 Istanbul, Turkey, Orcid ID: 0009-0007-0907-3862, msimsir@yildiz.edu.tr,
* Correspondence: eeisik@yildiz.edu.tr; +905393285829

Reference: Maximizing Total Net Profit for Traveling Salesman Problem with Profits Using Metaheuristic Algorithms**.** The European Journal of Research and Development,3(1), 46-59.

Abstract

Travelling Salesman Problem with profits (TSPP) is a special case of the general Travelling Salesman Problem, all nodes must not be visited, but profit is collected from visited nodes. It is a well-known NP-hard combinatorial optimization problem in the literature. Because of the problem's complexity, exact methods cannot find the global optimum solution to this problem, so many heuristic and metaheuristic solution techniques are studied to find a feasible solution in a reasonable time. In this research, two different metaheuristic algorithms, Simulated Annealing with Many-moves and Variable Neighborhood Search, are proposed to solve the TSPP. Proposed algorithms are tested with three different problem instances and compared in terms of the efficiency of algorithms.

Keywords: Arc routing with profits, traveling salesman problem with profits, metaheuristic algorithms, simulated annealing, variable neighborhood search

1. Introduction

The traveling salesman problem (TSP) is one of the extensively studied combinatorial optimization problems in the literature [1]. Also, TSP can be categorized as a subcategory of Vehicle Routing Problems (VRP), a generalization of visiting customers with more than one vehicle. In generic TSP, given a list of n nodes and distances between each pair of nodes. Then, a salesperson must visit all customers in the network once and return to the depot node while minimizing its total tour length. All customers must be served in the

network. The order-picking problem in warehouses, vehicle routing, computer wiring, X-Ray crystallography, and overhauling gas turbine engines are an example of real-life applications of TSPs [2]. As can be seen, the TSP application area is very broad and diverse in real life.

Traveling salesman problems with profits (TSPP) are a generalization of the TSP, where it is unnecessary to visit all network vertices [3]. In this research, we are going to solve the TSPP where the salesperson collects some profit for visiting each customer. The aim while solving TSPPs is finding the best subset of the given nodes, which provides the maximum total profit earned from visited customers while minimizing the total cost of the tour. As can be noticed, the problem is a bi-objective or multi-objective.

TSP has been widely studied in history, but it has been proven that TSP is one of the NP-hard problems in the literature [1]. As is well known, NP-hard problems cannot be solved in polynomial time. Especially, the problem becomes impossible to solve optimally with its growing size. As the previous studies stated, TSPP is also an NP-hard problem [3]. Due to the complexity of the problem, instead of exact algorithms, metaheuristic, and heuristic methods are studied to obtain a qualified solution within a reasonable time. Metaheuristic algorithms do not guarantee global optimality, but a feasible solution can be found within a reasonable time.

Traveling Salesman Problems with Profits (TSPP), one of the extensions of TSP, has attracted less attention from the literature compared to TSP but has still been studied extensively in the literature. The fact that the problem is a combinatorial optimization problem with the NP-hard structure is one of the main reasons for the increase in the number of studies on this subject. Many exact and heuristic methods are developed to solve the problem and its variations [3]. Zhu et al. [4] addressed the probabilistic version of the problem as multi-objective. The authors used the noise-aware evolutionary algorithm to solve the problem. Jozefowiez et al. [5] similarly handled the problem as multi-objective and aimed to produce Pareto optimal solutions. The authors introduced a hybrid meta-heuristic approach that combines ejection chain local search and multi-objective evolutionary algorithms in their work. Berube et al. [6] developed an exact ε-constraint solution method for bi-objective problems. The authors applied this developed ε-constraint method on TSPP and reported their results on benchmark instances and Pareto frontiers. Angelelli et al. [7] analyzed the complexity of TSPP and proposed polynomial approximation algorithms for some NP-hard cases in different versions of the problem. Lahyani et al. [8] focused on the multi-constrained version of TSPP and presented a unified solution approach based on the variable neighborhood search algorithm for the solution. The authors compared the system they developed in the study

with different methods in the literature. Zhang et al. [9] have defined a new version of TSPP considering stochastic customer presence. For the newly defined problem, the authors presented a nonlinear integer programming model and proposed a genetic algorithm for the solution. Jaszkiewicz [10] studied solving multi-objective problems and presented a Pareto local search algorithm. The authors applied the algorithm they developed on TSP and TSPP and compared their results with those obtained from different Pareto local search versions. Osicka et al. [11] focused on determining profits (or prizes) in TSPP and emphasized that customers can cooperate among themselves.

TSPP, by definition, has two objectives; minimization of total cost and maximization of total profit. The problem can be defined differently, depending on how these objectives are addressed. The problem is called the profitable tour problem when considering both objectives together minimizes the difference between the total profit and the total cost. In addition, the problem is defined as the orienteering problem in cases where the total profit maximization objective is considered as a constraint, and the total cost is minimized; defined as the prize-collecting TSP when the total cost minimization objective is considered as a constraint, and the total profit is tried to be maximized [3]. Unlike these definitions, in this study, TSPP is solved to maximize the total net profit.

TSPP is also examined under the general heading of routing problems with profits. In this scope, problems such as vehicle routing problems with profits, orienteering problems, and profitable tour problems are studied. However, since it is out of the content of this study, related studies are not mentioned in detail. Interested readers can visit different studies on the topics [12], [13].

The rest of the paper is organized as follows, Section 2 defines the solution approach for the given problem and introduces the datasets used to test the proposed metaheuristic algorithms. Section 3 represents the proposed algorithms' results. Section 4 discusses the efficiency and differences between the two proposed algorithms for TSPPs and concludes the paper by giving future research directions.

2. Materials and Methods

Let $G(N, E)$ be a graph where $N = (n_1, n_2, n_3, \dots n_i)$ is a set of i nodes, and E is the set of edges (undirected TSPs with profit). Let c_{ij} is the cost associated between node n_i and n_j, corresponding edge $e_{ij} \in E$. Let p_i is the price or profit that can be collectible from if $n_i \in N$ is visited. In this manner, nodes are customers to visit, and costs are the corresponding distances between each pair of nodes. The collectible price is the profit that will be collected from the visited customer. The depot location of the network is the n_1 and the salesman always starts their tour from the depot location and then visits their customers.

The objective of the problem is maximizing the total net profit, which is equal to the total profit earned from visited customers, less the total cost of the tour.

TSPP has an NP-hard structure like TSP. Studies on its complexity have revealed the difficulty of the problem. Therefore, as the size of the problem increases, the number of nodes increases and the solution times for exact solution methods increase exponentially. This situation reveals the need to find quality solutions quickly. This is where heuristic and metaheuristic algorithms come into play.

In this study, two metaheuristic algorithms are used for TSPP where total net profit is tried to be maximized. In the preliminary analysis of the problem for the dataset used, it is observed that the solution space contains many local optimums, making it hard to find the global optimum for search algorithms. In these cases, diversification should be increased, and local optimum points should be avoided. Figure 1 exemplifies local and global optimum points and indicates aspects of diversification and intensification. Simulated Annealing (SA), one of the first algorithms with an explicit strategy to avoid local optimum points, is used with different neighborhood structures to overcome this situation. Also, another algorithm that allows us to use many neighborhood structures Variable Neighborhood Search (VNS) is used to overcome this issue. The analyses show that VNS provides better results for the problems and instances under consideration.

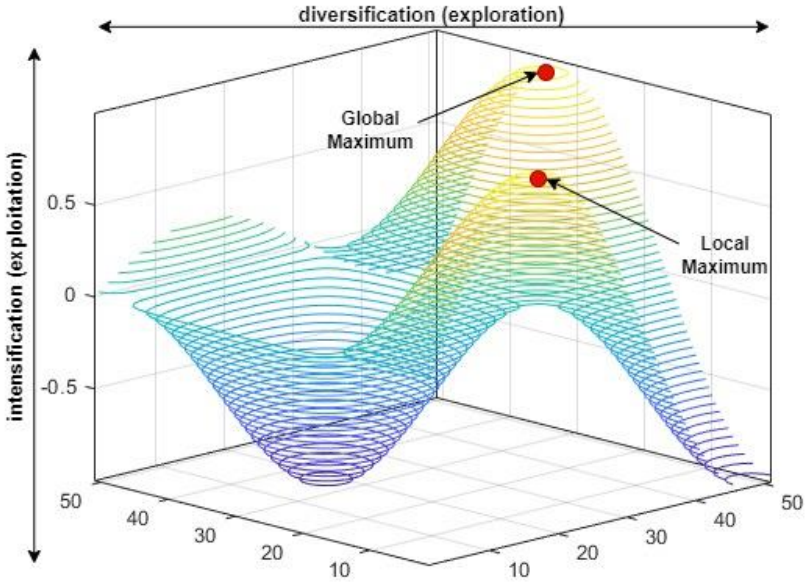

Figure 1: An example of local and global optima

2.1. Simulated Annealing (SA) with Many-moves

SA is a metaheuristic algorithm used for many optimization problems in the literature [14]. Algorithm nature is inspired real annealing procedure of alloys, with changing temperature certain alloys like metal, glass and etc. solidifies into a perfect crystalline structure. High quality of materials are produced by this physical process. SA main idea is depending on this annealing procedure. An initial solution is needed to start SA. SA creates a random candidate solution based on the neighbourhood structure using the current solution at each iteration, then decides whether to accept. In SA, better moves are always accepted, but worsening moves are accepted with a probability. A function depends on the temperature and changes in the evaluation function to control accepting a worsening solution. Based on this mechanism, the probability of accepting a worsening solution decreases when the temperature decreases. Accepting worsening solution mechanism of SA helps exploration, and it can also help the problem of being stuck in local optima. It allows the algorithm to explore the different areas in the search space, so it accepts bad solutions and hopes to find a good solution after that.

For the acceptance probability cooling schedule must be appropriately constructed. Starting temperature must be hot enough to allow the algorithm to move in search space, but if it is too hot, the algorithm may conduct a random search, so the cooling schedule is essential in terms of the quality of the SA. After trial and error, starting temperature of the SA is determined as 1000, and for the temperature decrement, geometric decrement is used. At each temperature, the algorithm runs for one epoch because the number of iterations at each temperature increases exponentially with the growing problem size. For the temperature decrement parameter β is chosen as 0.8 [15]. Lowering the temperature will take a longer time with higher values of β. On the other hand, lower values of β will let the temperature decrease too fast.

In SA, one neighborhood structure is used for creating a candidate solution, but in this problem, the tour length can be changed at each iteration. For TSP problems length of the tour is constant because all customers must be visited. For the TSPP, all customers must not necessarily be visited instead of TSP; this means that we cannot use constant tour length. The reason why SA with Many Moves is chosen instead of general SA, this algorithm diversification mechanism works better so search space can be explore quickly because a random function is chosen among given neighborhood structures for candidate solutions at each iteration. For example, as a neighborhood structure, only the swap operator is used it cannot improve solutions. More than one move, like insertion, swapping, inversing operators, which lead to change length of the tour at each iteration, can help improve the solutions' quality. This one is more suitable for this problem structure.

The insertion operator selects a random node among non-visited nodes and adds the selected node to a random point of the current solution. Adding one node, two, and three nodes are used to create a candidate solution. A simple example represented in Figure 2 Inversion operation randomly selects two nodes among visited nodes and then reverses the route between two nodes (Figure 3). Instead of the inversion operation, as seen in Figure 4, two-swap operators randomly select two nodes among visited nodes and then replace the location of only the selected two nodes, and the candidate solution is obtained. These operations allow us to search the close neighborhood of the current solution. The pseudocode of the proposed SA with Many Moves is given in Algorithm 1.

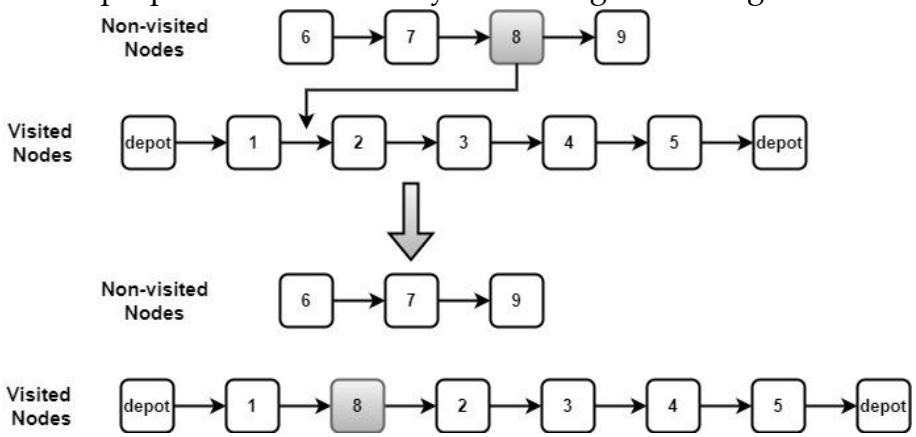

Figure 2: Insertion (Add) operator

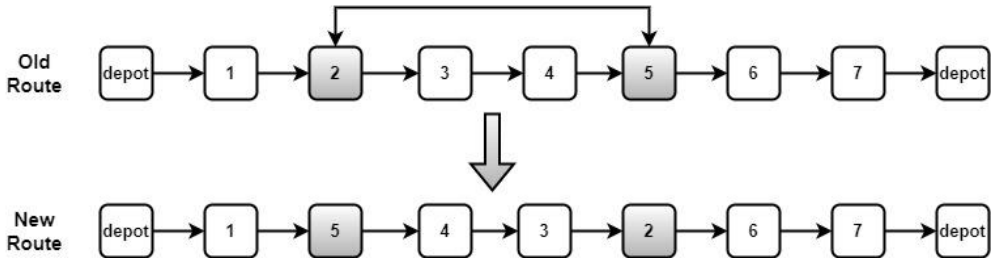

Figure 3: Inversion operator

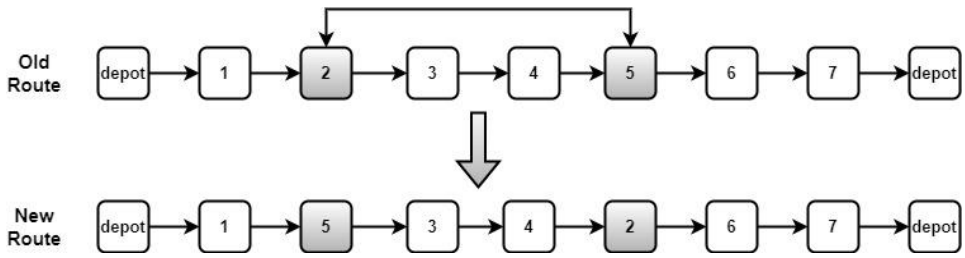

Figure 4: Swap operator

Algorithm 1: SA Algorithm with Many Moves for Solving TSPP

```
INPUT: t_max, HP, LP, Coord, functionlist
OUTPUT: The tour for that maximizes the total profit
BEGIN
  Read input files
  Calculate distance matrix using Coord
  Generate an initial solution X randomly
  Calculate the objective value of X using HP or LP
  current <- X
  f(current) <- f(X)
  L0<-1
  t0<-1000
  WHILE t < t_max
    FOR i 1 to L0
      candidate <- a randomly selected successor of current
      f(candidate) <- calculate the objective value of candidate
      delta <- f(candidate) – f(current)
      IF (delta > 0)
        current <- candidate
        f(current) <- f(candidate)
      ELSE (rand (0,1) => exp(-delta/t0)) // with prob accept worsening solution
        current <- candidate
        f(current) <- f(candidate)
      t0<-t0*β
      L0<-L0*α
    t <- CpuTime ()
  ENDWHILE
  Print current and f(current)
END
```

2.2. Variable Neighborhood Search (VNS)

In the preliminary analysis, we noticed that the problem has many local optima; traditional search algorithms are stuck with these local optima and cannot improve the solution. Similar problems arise when we start searching with a good solution using a constructive heuristic algorithm. For this reason, we used the Variable Neighborhood Search (VNS) algorithm, which is one of the basic algorithms that allow us to use different neighborhood structures simultaneously to scan the solution space comprehensively. The VNS algorithm basically includes the steps given in Algorithm 2 [16].

Algorithm 2: Steps of the basic VNS

```
INPUT: x, t_max, k_max
OUTPUT: x''
BEGIN
  WHILE t < t_max
    k <- 1
    REPEAT
      x' <- Shake (x, k)
      x'' <- LocalSearch (x')
      NeighborhoodChange (x, x'',k)
    UNTIL k = k_max
    t <- CpuTime ()
  ENDWHILE
  Print x
END
```

We used five neighborhood structures to scan the solution space within the VNS algorithm effectively. Since it is unnecessary to visit all points due to the nature of the problem, we used discrete encoding for the solution representation. This representation stores the visited points in the order they were visited. Since the algorithm works relatively fast, we used the total time instead of the number of iterations as the stopping condition. This way, we aimed to scan the solution space more comprehensively within the specified time.

The algorithm initially needs the maximum total duration (t_{max}) and the number of neighborhood structures (k_{max}), and the profit (*HP*, *LP*), coordinate (*Coord*) information of the points determined for the TSPP as an input. Here, we obtained the initial solution by generating a random solution according to the total number of nodes. So, as an initial, we accept a solution that assumes all points are visited.

For the shaking procedure, we chose four of the five neighborhood structures we used in the algorithm, and we ensured that one of them was randomly selected at each step. We did not use the Drop (or Deletion) operator in the shaking procedure as they narrowed the solution. If the objective value of the solution we obtained is better than the first solution, we continue with the same neighborhood structure. Otherwise, we start using the new neighborhood structure by increasing *k* by one.

We used the operators 2-Swap, 3-Opt, Drop (Deletion), Add (Insertion), and Replacement, respectively. Using the 2-Swap and 3-Opt operators, we aimed to find a

better tour among the solutions' points. As seen in Figure 5, the 3-Opt operator chooses three edges from the current tour, removes them, and combines the three parts to a current tour in the most profitable way to link them. There are eight (2^3) possible tours, and the operator chooses the best one according to the objective function value.

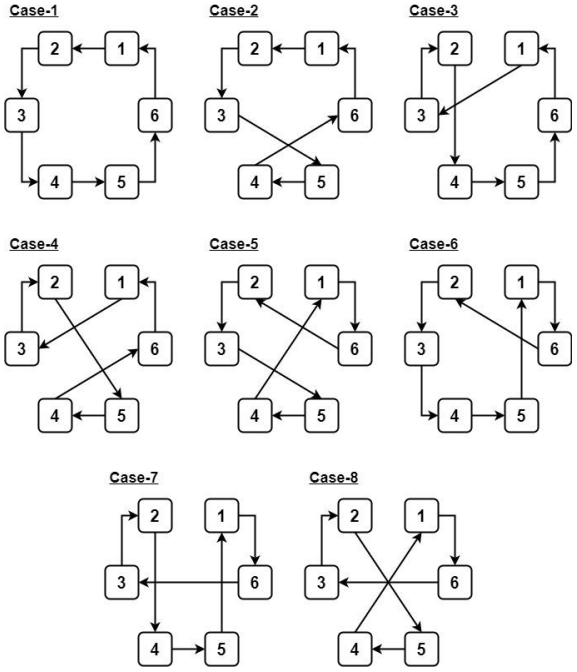

Figure 5: 3-Opt operator

The drop operator removes a random point from the current solution (Figure 6). On the other hand, the insertion operator allows a point not included in the solution to be added to a random point of the solution. Finally, the replacement operator replaces a point in the solution with a point not in the solution (Figure 7). With these operators, we aimed to find a more profitable tour by including the points not included in the current tour or removing an existing point from the solution. The pseudocode of the VNS algorithm for solving TSPP is given in Algorithm 3.

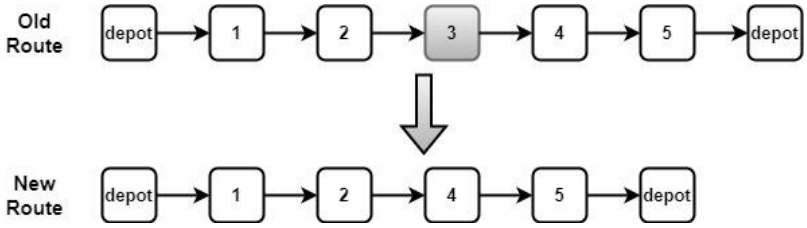

Figure 6: Deletion (Drop) operator

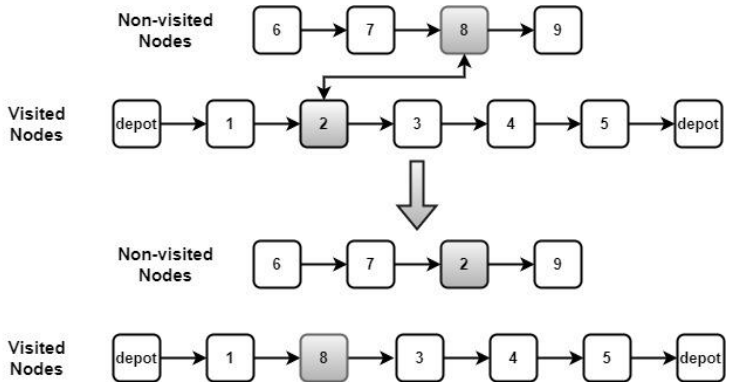

Figure 7: Replacement operator

Algorithm 3: VNS Algorithm for Solving TSPP

```
INPUT: t_max, k_max, HP, LP, Coord
OUTPUT: The tour for that maximizes the total profit
BEGIN
  Read input files
  Calculate distance matrix using Coord
  Generate an initial solution X randomly
  Calculate the objective value Z of X using HP or LP
  WHILE t < t_max
    k <- 1
    WHILE k < k_max
      r <- randi (k_max-1)
      X' <- Shake (X, r)
      X'' <- LocalSearch (X',k)
      Calculate the objective value Z' of X''
      IF Z' ≥ Z THEN
        X <- X''
        Z <- Z'
        k = 1
      ELSE
        k = k + 1
      ENDIF
    t <- CpuTime ()
  ENDWHILE
  Print X and Z
END
```

3. Result

For the analysis of the algorithms, three different datasets (eil51, eil76, and eil101) are used. All the instances include the x and y coordinates of the customers and potential collectible low (LP) and high profits (HP) from those customers. Different instances are used to see the effect of dimensionality on data with different customers and also to understand how low and high-profit values affect the algorithms' solution quality. The first customer indexed by "0" is the depot location, i.e., the salesperson starts its tour from

this location. Therefore, the number of customers equals 50, 75, and 100 for eil51, eil76, and eil101, respectively. Hence, the input data consists of the customer locations and the profits associated with each customer.

Termination criteria of the algorithms are determined as CPU time to compare the performances of the algorithms. Although the conversion speed of the algorithm varies according to the problem size, we ran the algorithm for 180 seconds for each problem size to establish a standard. The Euclidean distance between the nodes is calculated using the x and y coordinate information. We rounded all the Euclidean distances to two decimal points. The results of the algorithms are represented in Table 1.

Table 1: Comparison of proposed metaheuristic algorithms

Instances	SA with Many-moves			VNS			Gap(%)
	Best Objective Value	#nodes visited	CPU Time (s)	Best Objective Value	#nodes visited	CPU Time (s)	
eil51-LP	31.85	18	180	50.25	21	180	36.62
eil51-HP	118	32	180	159.91	40	180	26.21
eil76-LP	37.47	35	180	53.06	34	180	29.38
eil76-HP	329	57	180	380.66	74	180	13.57
eil101-LP	196	56	180	245.54	70	180	20.18
eil101-HP	943.41	88	180	1504.3	98	180	37.29

As seen in Table 1, VNS provides better results than SA with Many-moves regarding objective function value. In addition, in most problem instances, more nodes are visited in the solution obtained with VNS, while fewer nodes are visited only for the eil76-LP instance. However, even in this case, the total net profit value obtained is higher in solutions found with VNS. This shows that some nodes are more advantageous than others regarding the net profit value. The Gap(%) column in Table 1 shows the difference between the objective values obtained by both methods and is calculated with the formula given in Equation (1).

$$Gap(\%) = \frac{Best_{BVNS} - Best_{SA}}{Best_{BVNS}} * 100 \quad (1)$$

When the Gap(%) column is examined, it is seen that the gap value generally decreases as the problem size increases. This situation can be interpreted as SA with Many-moves may be more suitable for large-sized instances in itself. Another considerable result is that the gap value is less in instances with low profit than in instances with high profit. This

situation can be interpreted as the increase in the general profit values increasing the solution options, and it is easier to reach good solutions.

4. Discussion and Conclusion

Due to their complexity and many real-world applications, TSP and its extensions are among the most studied problems in the literature. Although only one of the extensions, TSPP, has received less attention from the literature compared to other versions, it plays an important role in solving real-life problems such as tourist trip design, steel rolling mill scheduling, or inventory routing. Although exact solutions have been developed for TSPP, the literature generally focuses on approximate approaches to obtain quick quality solutions. Heuristic and metaheuristic algorithms are among the most preferred methods.

In this study, TSPP, which maximizes the total net profit by going beyond the definitions made in the literature, is discussed. After emphasizing the difference between the general information about TSPP and the definition of the problem in the study compared to the ones in the literature, two different approaches are proposed for the solution of the problem. As a result of the analysis made with the dataset in question, it was determined that the solution space has many local optima, so algorithms that increase the amount of diversification were used.

SA, which is one of the first algorithms focusing on avoiding local optima, was the first choice because of these features. However, SA with Many-moves was proposed for the solution of TSPP by using different neighborhood structures to increase diversification. The use of different neighborhood structures together highlighted the VNS algorithm. For this reason, the second method proposed for the solution of TSPP within the scope of the study was determined as VNS.

The proposed solution methods were tried for profit values of different sizes and levels, and the results were reported. According to these results, it is seen that VNS provides better objective values than SA with Many-moves. This situation was interpreted as VNS's ability to carry out exploration in larger areas thanks to the shaking procedure. In future studies, the problem can be addressed in larger sizes, and similar analyses can be repeated on other versions. In addition, population-based or swarm-based algorithms can also be applied to the solution of TSPP.

References

[1] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 2, pp. 231–247, 1992, doi: 10.1016/0377-2217(92)90138-Y.

[2] R. Matai, S. Singh, and M. Lal, "Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches," *Travel. Salesm. Probl. Theory Appl.*, 2010, doi: 10.5772/12909.

[3] D. Feillet, P. Dejax, and M. Gendreau, "Traveling salesman problems with profits," *Transp. Sci.*, vol. 39, no. 2, pp. 188–205, 2005, doi: 10.1287/trsc.1030.0079.

[4] B. Zhu, J. Suzuki, and P. Boonma, "Solving the Probabilistic Traveling Salesperson Problem with Profits (pTSPP) with a Noise-aware Evolutionary Multi-objective Optimization Algorithm," 2011.

[5] N. Jozefowiez, F. Glover, and M. Laguna, "Multi-objective meta-heuristics for the traveling salesman problem with profits," *J. Math. Model. Algorithms*, vol. 7, no. 2, pp. 177–195, 2008, doi: 10.1007/s10852-008-9080-2.

[6] J. F. Bérubé, M. Gendreau, and J. Y. Potvin, "An exact ε-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits," *Eur. J. Oper. Res.*, vol. 194, no. 1, pp. 39–50, 2009, doi: 10.1016/j.ejor.2007.12.014.

[7] E. Angelelli, C. Bazgan, M. G. Speranza, and Z. Tuza, "Complexity and approximation for Traveling Salesman Problems with profits," *Theor. Comput. Sci.*, vol. 531, pp. 54–65, 2014, doi: 10.1016/j.tcs.2014.02.046.

[8] R. Lahyani, M. Khemakhem, and F. Semet, "A unified matheuristic for solving multi-constrained traveling salesman problems with profits," *EURO J. Comput. Optim.*, vol. 5, no. 3, pp. 393–422, 2017, doi: 10.1007/s13675-016-0071-1.

[9] M. Zhang, J. Qin, Y. Yu, and L. Liang, "Traveling salesman problems with profits and stochastic customers," *Int. Trans. Oper. Res.*, vol. 25, no. 4, pp. 1297–1313, 2018, doi: 10.1111/itor.12310.

[10] A. Jaszkiewicz, "Many-Objective Pareto Local Search," *Eur. J. Oper. Res.*, vol. 271, no. 3, pp. 1001–1013, 2018, doi: 10.1016/j.ejor.2018.06.009.

[11] O. Osicka, M. Guajardo, and K. Jörnsten, "Cooperation of customers in traveling salesman problems with profits," *Optim. Lett.*, vol. 14, no. 5, pp. 1219–1233, 2020, doi: 10.1007/s11590-019-01429-6.

[12] C. Archetti and M. G. Speranza, "Chapter 12: Arc Routing Problems with Profits," in *Arc Routing*, pp. 281–299.

[13] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering Problem: A survey of recent variants, solution approaches and applications," *Eur. J. Oper. Res.*, vol. 255, no. 2, pp. 315–332, 2016, doi: 10.1016/j.ejor.2016.04.059.

[14] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, 1983, [Online]. Available: https://www.jstor.org/stable/1690046.

[15] T. L. Friesz, H. J. Cho, N. J. Mehta, R. L. Tobin, and G. Anandalingam, "Simulated

annealing approach to the network design problem with variational inequality constraints," *Transp. Sci.*, vol. 26, no. 1, pp. 18–26, 1992, doi: 10.1287/trsc.26.1.18.

[16] P. Hansen and N. Mladenović, "Variable neighborhood search," *Handb. Heuristics*, vol. 1–2, pp. 759–787, 2018, doi: 10.1007/978-3-319-07124-4_19.