Advances in Social Science, Education and Humanities Research, volume 550
Proceedings of the 1st International Conference on Mathematics and Mathematics Education (ICMMED 2020)

Study of the Brand and Bound Algorithm Performance on Traveling Salesman Problem Variants

Sapti Wahyuningsih[1,*] Dwi Retno Sari[2]

[1] *Universitas Negeri Malang, Indonesia*
[2] *Universitas Negeri Malang, Indonesia*
[*]*Corresponding author. Email: sapti.wahyuningsih.fmipa@um.ac.id*

ABSTRACT
Traveling salesman problem (TSP) can be applied to distribution problems, namely determining the minimum route from the depot to all customers exactly once and back to the depot. Some constraints can be added to the problem such as time constraints, additional salesmen on the route that is passed, the need for an order of delivery, and passing one-way road. This article will discuss TSP variants developed from basic TSP, namely TSPTW, MTSP, TSPPC, and ATSPTW. The differences in formulations of these variants are described and the branch and bound algorithm is used to solve them. There are three main steps to the branch and bound algorithm namely the initialization stage to obtain the initial solution, the process stage and solution improvement, and the determination of the optimum solution. Identification of the similarities and differences in the application of the branch and bound algorithm steps on these variants are discussed in this article. An example is given of the application of the branch and bound algorithm to the four variants of 8 customers and one depot. The results of the application examples are depicted on a graph in order of the minimum total distance, namely ATSPTW, TSPTW, TSPPC, and MTSP.

Keywords: *the branch and bound algorithm, ATSPTW, TSPTW, TSPPC and MTSP.*

1. INTRODUCTION

Traveling Salesman Problem (TSP) is a problem for a salesman to determine the minimum travel distance to visit all cities exactly once starting and ending in the same city. In real problems, more complex problems are found, for example, there is a time limit for delivery, one-way road, more than one salesman, a reduction in customers on the way, and a delivery order requirement. New variants of the existing basic TSP developed. Several previous studies that removed baseline TSP can be seen in [1], [2], [3], [4], [5], and [6].

TSP variants were developed not only for solving complex real problems but also for obtaining better results with the addition of certain constraints. Traveling Salesman Problem with Time Window (TSPTW) is a TSP variant with time window time constraints. Studies on TSPTW can be seen in [7], [8, p.], And [9]. Asymmetric Traveling Salesman Problem (ATSP) variant of TSP development with the return and return routes between two cities is not necessarily the same. ATSP discussion can be seen in [10] and [11]. Variant modifications developed from ATSP and TSPTW can give rise to new variants, namely ATSPTW. Previous research that discusses ATSPTW can be seen in [12].

Multiple Traveling Salesman Problem (MTSP) is a TSP variant with additional constraints on the number of salesmen more than one. Previous research on MTSP studies can be seen in [13], [14], and [15]. Traveling Salesman Problem with Precedence Constraints (TSPPC) is a TSP variant with constraints on the order of delivery requirements. Previous studies on TSPPC can be seen in [16], [17], and [18]. Studies to determine the differences in the formulations of these variants are needed because it will facilitate the application of an algorithm.

Various algorithms can be used to solve TSP variants. In article [17] the Branch-and-bound algorithm for the Precedence Constrained Generalized Traveling Salesman Problem. In article [12] discussed the branch and bound algorithm in ATSPTW, an article [19] discussed the branch and bound algorithm in the asymmetric traveling salesman problem with replenishment arcs. Articles [20] and [21] examine the branch and cut algorithm.

The branch and the bound algorithm have exact steps that can be adjusted for different variants of TSP. This article will examine the application of the branch and bound algorithm on ATSPTW, MTSP, TSPPC, and TSPTW. Study on the different formulations of ATSPTW, MTSP, TSPPC, and TSPTW, and will identify the specificity of the branch and bound algorithm steps for the TSP variant problem. To see the differences in the results of implementing the branch and bound algorithm on ATSPTW, MTSP, TSPPC, and TSPTW, examples of its application are given.

2. RESULT AND DISCUSSION

The formulations of the ATSPTW, MTSP, TSPPC, and TSPTW variants were developed from the formulations on the basic TSP with the addition of different constraints. Additional salesman time window constraints in a customer and limits not exceeding the specified time limit are added to the ATSPTW and TSPTW variants. The ATSPTW variant uses an asymmetric graph, namely a graph with the weight *i* to *j* is not necessarily the same as weight *j* to *i*. For other variants using a symmetric graph. The MTSP variant uses several salesmen for distribution (for example m salesmen), while for other variants it uses a single salesman. In the TSPPC variant, precedent constraints are added, namely the existence of a certain order in delivery. The following is the formulation of additional TSP variant constraints:

i). Time to start salesman service to customers *j*
$S_i + c_{ij} - M(1 - x_{ij}) \leq S_j,$
ii). Service (time window) at customers i,
$a_i \leq S_i \leq b_i, \ \forall i \in V$
iii). Graph type-directed $(c_{ij} \neq c_{ji}, i \neq j$ or Symmetric $(c_{ij} = c_{ji})$
iv). m salesman leaving the depot $\sum_{i=1}^{n} x_{1j} = m$

m salesman arrives back at the depot $\sum_{j=1}^{n} x_{i1} = m$
v). All trips meet precedent constrains

$\sum_{i\in S-\{1\}} \sum_{j\in V-S} x_{ij} \geq 1 \quad \forall S \subset V, 1 \in S, \forall q \in S, \forall r \notin S, q \prec r$
q<r represents the precedent constrains statement

Information:

S_i Service time at customer *i*

c_{ij} time from customer *i* to customer *j*
a_i time window shows the initial limit of service to customers *i*
b_i window shows the end of service limit at customer *i*
M positive constant whose value is relatively large.
The branch and bound algorithm steps to solve the ATSPTW, MTSP, TSPPC, and TSPTW variants are as follows: in the MTSP variant, the point grouping stage is carried out using the k-means algorithm. The k-means algorithm has a high enough accuracy concerning object size, so this algorithm is relatively more scalable and efficient for processing large numbers of objects. Each group obtained from the k-mean algorithm was given the initialization stage treatment to obtain the initial solution. In [22] discussed the k-optimal algorithm for the traveling-salesman problem. In general, the steps for the branch and bound algorithm for all TSP variants are as follows.

Initialization stage to obtain initial solutions
Step 1. Determine a route in the order of travel from start to finish with a limit on the minimum value of the objective function and search for various possible travel routes. This limit is indicated by f_u, and go to step 2.
Step 2. Process of repairing the solution.

- In the TSPPC variant, branching is made according to precedence constraints. Make the initial fork by $x_1 = 1$ for each city (point $j = 2,3,\dots,n$ (unless $c_{ij} = M$ an impossible route). Calculate the lowest limit f_L on the minimum value of the objective function in each city (point) as follows. From the initial data, delete the first row and column j to. Next, set Determine the result of the assigned problem solving and add a value (f) to c_{1i} $f_L = c_{1j} + f$. If $f_L < f_U$ to the next step (step b).
- If no city is visited then the current best solution is optimal. If not, select the point with the smallest f_L and create a new branch by $x_{jk} = 1$ for each k city (point) that has not been previously visited on part of the trip. Continue at step c.
- Create a constraint f at each point by removing row j and column k from the troubleshooting at the current point above the existing point, setting $c_{kj} = n$ adding the f value to c_{jk} and all the previous values in the part of the trip.

Step 3. Optimum conditions

Step 2 above is repeated until there is no branching point, then a certain minimum lower limit value is obtained which is the best solution produced by the Branch and bound algorithm. Then determine the time from multiplying distance to speed.

To see the differences and similarities in the settlement steps with the branch and bound algorithm, it can be seen in Table 1. below.

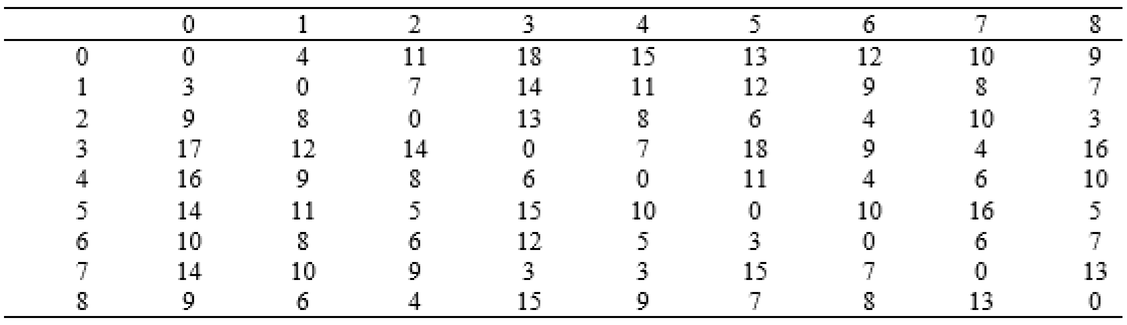
Table 1. Summary of the steps for the branch and bound algorithm

Stage	ATSPTW, TSPPC, TSPTW	MTSP
Inisialisasi	Determine a route with a complete trip sequence, create an Upper bound at the minimum value of the objective function by looking for various possible trips. This limit is indicated by f_u *bound*	Initialization stage of k-means algorithm grouping. Determine the cluster for a salesman. Each cluster route is determined with a complete trip sequence, make the upper bound (Upper bound) at the minimum value of the objective function by looking for various possible trips. This limit is indicated by f_u
Process (steps *Branch and bound*)	Determine the branches for each point, only at TSPPC the branches according to the delivery order Determine the lower bound, namely the minimum value of the route obtained from the fork If there is a lower bound that is smaller than the upper limit, then branch off the points.	Determine the branch for each point Determine the lower limit, namely the minimum value of the route obtained from the fork If there is a lower bound that is smaller than the upper limit, then branch off the points.
Optimum Condition	The optimum condition is reached when all Branch and bound structures have been passed and produce the best solution that leads to a fixed value.	The optimum condition is reached when all Branch and bound structures have been passed and produce the best solution that leads to a fixed value.

Example of Application of Branch and bound Algorithm for ATSPTW

For example, 1,2,3,4,5,6,7,8 customers must be visited by salesmen who depart from 0 (depot). How to find the distance and minimum travel time for the salesman to visit each customer who is visited exactly once and the salesman returns to the depot with an average vehicle speed of 60 km/hour and the service time for each customer or depot is the same, which is 10 minutes? The following will be given in Table 2 the distance between depots to each customer and between customers.

Table 2. Distance Between Depots to Each Customer and Between Customers



	0	1	2	3	4	5	6	7	8
0	0	4	11	18	15	13	12	10	9
1	3	0	7	14	11	12	9	8	7
2	9	8	0	13	8	6	4	10	3
3	17	12	14	0	7	18	9	4	16
4	16	9	8	6	0	11	4	6	10
5	14	11	5	15	10	0	10	16	5
6	10	8	6	12	5	3	0	6	7
7	14	10	9	3	3	15	7	0	13
8	9	6	4	15	9	7	8	13	0

Information:

Line 0 shows the distance from the depot to each customer, the depot to customer 1 distance is 4 km. Line 1 shows the distance from customer 1 to the depot and other customers, the distance of customer 1 to the depot is 3 km, the distance of customer 1 to customer 2 is 7 km, and so on.

Following are the steps to solve ATSPTW using the Branch and bound algorithm.

Step 1. Initial Solution Initial Phase

Table 3. The calculation results

	0	2	3	4	5	6	7	8
1	M	7	14	11	12	9	8	7
2	9	M	13	8	6	4	10	3
3	17	14	M	7	18	9	4	16
4	16	8	6	M	11	4	6	10
5	14	5	15	10	M	10	16	5
6	10	6	12	5	3	M	6	7
7	14	9	3	3	15	7	M	13
8	9	4	15	9	7	8	13	M

So the solution is

$x_{12} = x_{28} = x_{85} = x_{54} = x_{46} = x_{67} = x_{73} = x_{30}$

with value

$f = 7 + 3 + 7 + 10 + 4 + 6 + 3 + 17 = 57$,

so $f_L = C_{01} + f = 4 + 57 = 61$, so $f_L < f_U$ and vertex 1 can be measured.

The branch point and the next iteration is done in the same way so that there are no more branching points.

Step 3. Optimum condition

There is no branching point so that the optimal route solution is obtained $0 - 7 - 3 - 4 - 6 - 5 - 2 - 8 - 1$ with a distance of 44 km and a time of 2,233 hours.

Example of Application of Branch and bound Algorithm for MTSP

A company has 2 employees as salesmen. The two salesmen will distribute goods to 8 customers who are in different locations for the customer and company positions shown in Table 4. The company aims for each salesman to serve different customers and take a trip to the customer's location by returning to the company, quickly and efficiently. The problem is how to find the minimum route distance to the salesman so that the company's goals can be achieved.

Table 4. Point Coordinates Data

Point	Coordinate X	Coordinate Y
v_0	2	1
v_1	6	3
v_2	8	10
v_3	20	5
v_4	15	10
v_5	5	15
v_6	12	8
v_7	16	4
v_8	5	10

Based on the data in Table 4, each point is represented by coordinates (x, y) in the cartesian coordinate plane. For example, point v_0 is the position of the company with coordinates (2,1), v_1 is the customer with coordinates (6,3), point v_2 is customer 2 with coordinates (8,10), and so on. The following is the solution to the above MTSP problems: in grouping and stages in the branch and bound algorithm, data on the distance between points is required. So that to get distance data between points can be obtained from the calculation of the following formula:

$$\|v_{ij}\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} : i = 1, \dots, n; j = 1, \dots, k$$

As an example of calculating point distance v_4 dan v_5 , point distance v_6 dan v_7 are as follows:

$$\|v_{45}\| = \sqrt{(15-5)^2 + (10-15)^2} = \sqrt{10^2 + (-5)^2} = \sqrt{125} = 11{,}180$$

After obtaining the point distance, the point grouping will be sought using the K-Means algorithm, then the point grouping will be obtained v_1, v_3, v_6, dan v_7 included in *cluster* c_1 while the point v_2, v_4, v_5, dan v_8 included in *cluster* c_2.

To find a route solution for each cluster formed, the next stage for each cluster is resolved using a branch and bound algorithm. By iterating the branches, the minimum value of the branching will be obtained. Iteration is carried out until there are no more branching points. The optimum solution is obtained for *cluster* c_1 is $v_0 - v_7 - v_3 - v_6 - v_1 - v_0$ with distances 39.267 km. Optimal solution on *cluster* c_2 obtained the result 43,484 km, so the total route of the two salesmen is 82,484 km.

Examples of Application of Branch and Bound Algorithms for TSPPC and TSPTW

If there are cities 1,2,3,4,5,6,7,8 that should be visited by salesmen and 0 indicates the starting city. The distance between cities is given in Table 5. The distance data is an asymmetrical case in ATSPTW for which a solution will be sought with the TSPPC and TSPTW cases.

Table 5. Distance data between cities

City	0	1	2	3	4	5	6	7	8
0	0	4	11	18	16	14	12	14	9
1	4	0	7	14	11	12	8	10	7
2	11	7	0	13	7	6	4	10	3
3	18	14	13	0	7	18	9	4	16
4	16	11	7	7	0	11	4	6	10
5	14	12	6	18	11	0	10	16	5
6	12	8	4	9	4	10	0	6	7
7	14	10	10	4	6	16	6	0	13
8	9	7	3	16	10	5	7	13	0

The TSPPC case: How to find the minimum distance traversed by a salesman with each city visited exactly once with the provisions of the cities that must be visited first (precedence constraints), which is shown in the directed graph (digraph) in Figure 1.

TSPTW case: How to find the minimum distance and time traveled by a salesman with an average vehicle speed of 60 km/hour and the service time is assumed to be the same for each city, namely 10 minutes?

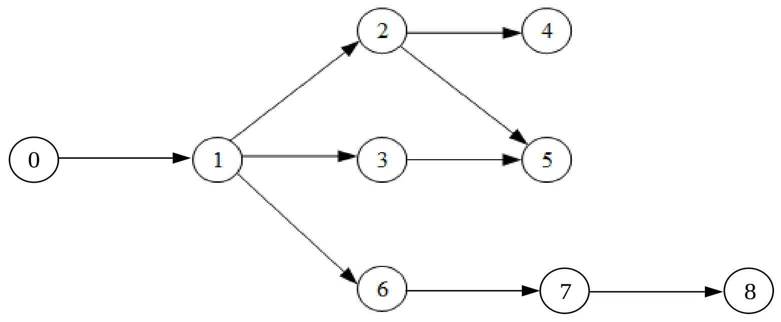


Figure 1. Digraph with 9 vertices and 9 arcs

The following will be resolved for the TSPPC case with precedence constraints in Figure 1 and distance data for calculating boundary values in the branch and bound algorithm.

Step 1. Initial Solution Initial Phase

The solution to the specified problem is $x_{01} = x_{12} = x_{23} = x_{34} = x_{45} = x_{56} = x_{67} = x_{78} = x_{80}$
possible with the corresponding values in the distance table i 4 + 7 + 13 + 7 + 11 + 10 + 6 + 13 + 9 = 80 so that it is set to $f_u = 80$

Step 2. Process

The formation of branches is adjusted to the rules of delivery. Create a branch point 0, then make a branch point at the minimum from branch 0, do the branching until it's finished. When there is no branching point, the solution of the minimum boundary will be obtained.

Step 3. Optimum conditions

The optimum solution with precedent constraints is 0-1-2-6-7-3-4-5-8-0 with a distance of 57 km.

The following will be resolved for the TSPTW case with distance data to calculate boundary values in the branch and bound algorithm.

Step 1. Initial Solution Initial Phase

$x_{01} = x_{12} = x_{23} = x_{34} = x_{45} = x_{56} = x_{67} = x_{78} = x_{80}$.
The solution to the specified problem is is possible with the appropriate values in the distance table 4 + 7 + 13 + 7 + 11 + 10 + 6 + 13 + 9 = 80 so that set $f_u = 80$

Step 2. Process

Create a branch point 0, then make a branch point at the minimum from branch 0, do the branching until it's finished. When there is no branching point, the solution of the minimum boundary will be obtained.

Step 3. Optimum conditions

The optimum solution is 0-7-3-4-6-2-5-8-1-0 with a distance of 55 km and a travel time of 2.42 hours.

From the example of the application of the branch and bound algorithm on the TSP variant, the results obtained by the analysis for ATSPTW have a minimum total distance due to the existence of other roads that are different and shorter or modeled with asymmetric digraphs. MTSP has more than one salesman so that the distribution of goods can be faster but the number of distances is greater because the distance traveled by each salesman is added up. The TSPPC has rules in order so the resulting solutions are limited to those rules. Meanwhile, TSPTW has time constraints with symmetric graph models. The following is given in Figure 2 a graph of the results of the application of the branch and bound algorithm for each TSP variant.

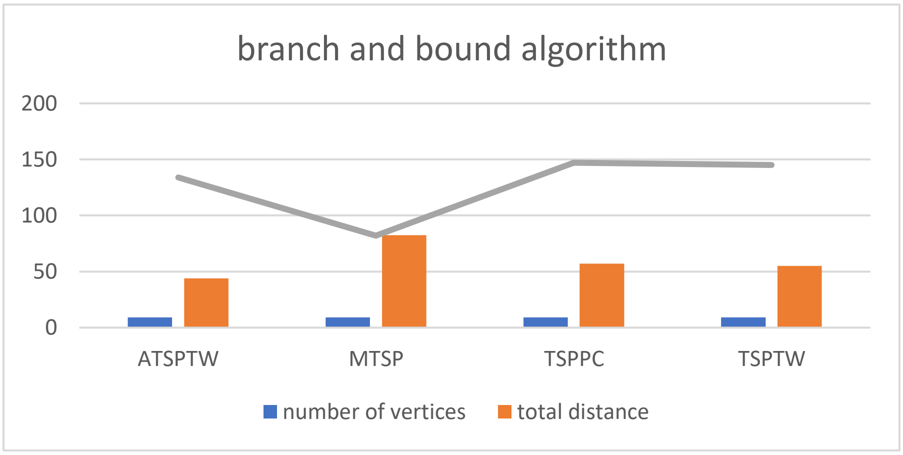


Figure 2. The results of the graph of the application of the branch and bound algorithm

The sequence of calculation results of the application of the TSP variant based on the smallest total distance is ATSPTW, TSPTW, TSPPC, MTSP. In the graph the number of customers and the depot shows the number of vertices, namely 9 vertices, the total distance is obtained from the total distance traveled by the salesman, for the total time on the graph is obtained from the conversion of distance divided by speed and converted into minutes.

3. CONCLUSION

The formulations of the ATSPTW, MTSP, TSPPC, and TSPTW variants have the same objective function. The constraint formulation of each variant is that the salesman arrives and leaves each customer exactly once. Additional constraints for ATSPTW and TSPTW are customer service time and time window, ATSPTW with asymmetric graph model and TSPTW graph symmetric, MTSP with more than one salesman, TSPPC all trips meet precedent constrains relationship.

The specificity of the branch and bound algorithm steps to solve the TSP variant is that in the MTSP before the initialization stage there is a grouping of customers according to the number of salesmen who will distribute and in the TSPPC the branching process is adjusted according to the order of delivery. The analysis of the application of the TSP variant is that ATSPTW has time constraints and the distance of round-trip routes which are not necessarily the same or modeled by asymmetric graphs. MTSP has more than one salesman so that the distribution of goods can be faster but the number of distances is greater because the distance traveled by each salesman is added up. TSPPC has rules in order so the resulting solutions are limited to those rules. Meanwhile, TSPTW has time constraints with symmetric graph models.

The ATSPTW, MTSP, TSPPC, and TSPTW variants can be developed in the presence of traffic congestion factors. Because in real conditions traffic jams cannot be avoided, even with the increasing number of private vehicles.

ACKNOWLEDGMENTS

This article is a part of research funded by PNBP UM, contract no: 4.3.476/UN32.14.1/LT/2020. Thank you to Universitas Negeri Malang for Funding the Research.

REFERENCES

[1] D. Nicola, R. Vetschera, and A. Dragomir, ‘Total distance approximations for routing solutions’,

Computers & Operations Research, vol. 102, pp. 67–74, Feb. 2019, doi: 10.1016/j.cor.2018.10.008.

[2] X. Wang, B. Golden, and E. Wasil, ‘A Steiner Zone Variable Neighborhood Search Heuristic for the Close-Enough Traveling Salesman Problem’, *Computers & Operations Research*, vol. 101, pp. 200–219, Jan. 2019, doi: 10.1016/j.cor.2018.07.023.

[3] I. Khan and M. K. Maiti, ‘A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem’, *Swarm and Evolutionary Computation*, vol. 44, pp. 428–438, Feb. 2019, doi: 10.1016/j.swevo.2018.05.006.

[4] S. S. Choong, L.-P. Wong, and C. P. Lim, ‘An artificial bee colony algorithm with a Modified Choice Function for the traveling salesman problem’, *Swarm and Evolutionary Computation*, vol. 44, pp. 622–635, Feb. 2019, doi: 10.1016/j.swevo.2018.08.004.

[5] Z. A. Aziz, ‘Ant Colony Hyper-heuristics for Travelling Salesman Problem’, *Procedia Computer Science*, vol. 76, pp. 534–538, 2015, doi: 10.1016/j.procs.2015.12.333.

[6] P. Venkatesh, G. Srivastava, and A. Singh, ‘A general variable neighborhood search algorithm for the k-traveling salesman problem’, *Procedia Computer Science*, vol. 143, pp. 189–196, 2018, doi: 10.1016/j.procs.2018.10.375.

[7] A. Montero, I. Méndez-Díaz, and J. J. Miranda-Bront, ‘An integer programming approach for the time-dependent traveling salesman problem with time windows’, *Computers & Operations Research*, vol. 88, pp. 280–289, Dec. 2017, doi: 10.1016/j.cor.2017.06.026.

[8] M. López-Ibáñez, C. Blum, J. W. Ohlmann, and B. W. Thomas, ‘The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization’, *Applied Soft Computing*, vol. 13, no. 9, pp. 3806–3815, Sep. 2013, doi: 10.1016/j.asoc.2013.05.009.

[9] I. Kara and T. Derya, ‘Formulations for Minimizing Tour Duration of the Traveling Salesman Problem with Time Windows’, *Procedia Economics and Finance*, vol. 26, pp. 1026–1034, 2015, doi: 10.1016/S2212-5671(15)00926-0.

[10] V. Mak and N. Boland, ‘Facets of the polytope of the asymmetric travelling salesman problem with replenishment arcs’, *Discrete Optimization*, vol. 3, no. 1, pp. 33–49, Mar. 2006, doi: 10.1016/j.disopt.2005.10.003.

[11] U. Boryczka and K. Szwarc, ‘The Harmony Search algorithm with additional improvement of harmony memory for Asymmetric Traveling Salesman Problem’, *Expert Systems with Applications*, vol. 122, pp. 43–53, May 2019, doi: 10.1016/j.eswa.2018.12.044.

[12] A. Arigliano, G. Ghiani, A. Grieco, E. Guerriero, and I. Plana, ‘Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm’, *Discrete Applied Mathematics*, vol. 261, pp. 28–39, May 2019, doi: 10.1016/j.dam.2018.09.017.

[13] P. Kitjacharoenchai, M. Ventresca, M. Moshref-Javadi, S. Lee, J. M. A. Tanchoco, and P. A. Brunese, ‘Multiple traveling salesman problem with drones: Mathematical model and heuristic approach’, *Computers & Industrial Engineering*, vol. 129, pp. 14–30, Mar. 2019, doi: 10.1016/j.cie.2019.01.020.

[14] C. Defryn and K. Sörensen, ‘Multi-objective optimisation models for the travelling salesman problem with horizontal cooperation’, *European Journal of Operational Research*, vol. 267, no. 3, pp. 891–903, Jun. 2018, doi: 10.1016/j.ejor.2017.12.028.

[15] M. Cornu, T. Cazenave, and D. Vanderpooten, ‘Perturbed Decomposition Algorithm applied to the multi-objective Traveling Salesman Problem’, *Computers & Operations Research*, vol. 79, pp. 314–330, Mar. 2017, doi: 10.1016/j.cor.2016.04.025.

[16] 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, Aug. 2002, doi: 10.1016/S0377-2217(01)00227-2.

[17] R. Salman, F. Ekstedt, and P. Damaschke, ‘Branch-and-bound for the Precedence Constrained Generalized Traveling Salesman Problem’, *Operations Research Letters*, vol. 48, no. 2, pp. 163–166, Mar. 2020, doi: 10.1016/j.orl.2020.01.009.

[18] A. Chentsov, M. Khachay, and D. Khachay, ‘Linear time algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem’, *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 651–655, 2016, doi: 10.1016/j.ifacol.2016.07.767.

[19] V. Mak and N. Boland, ‘Polyhedral results and exact algorithms for the asymmetric travelling

salesman problem with replenishment arcs', *Discrete Applied Mathematics*, vol. 155, no. 16, pp. 2093–2110, Oct. 2007, doi: 10.1016/j.dam.2007.05.014.

[20] G. Lera-Romero and J. J. Miranda-Bront, 'A branch and cut algorithm for the time-dependent profitable tour problem with resource constraints', *European Journal of Operational Research*, p. S0377221719305788, Jul. 2019, doi: 10.1016/j.ejor.2019.07.014.

[21] M. Barbato, R. Grappe, M. Lacroix, and R. Wolfler Calvo, 'Polyhedral results and a branch-and-cut algorithm for the double traveling Salesman problem with multiple stacks', *Discrete Optimization*, vol. 21, pp. 25–41, Aug. 2016, doi: 10.1016/j.disopt.2016.04.005.

[22] L. Zhang, H. Li, L. Meng, and J. Wang, 'Ordering of high-density markers by the k-Optimal algorithm for the traveling-salesman problem', *The Crop Journal*, vol. 8, no. 5, pp. 701–712, Oct. 2020, doi: 10.1016/j.cj.2020.03.005.