Solving Multi-Pickup and Delivery Problem with Time Windows using Ant Colony Optimization

Thuong Thanh Tran[1,2], Maria Art Antonette Clariño[1]
[1]*Institute of Computer Science, University of the Philippines Los Baños, Philippines*
[2] *Thai Nguyen University, Vietnam.*
ttran@up.edu.ph

***Abstract* — This paper presents a meta-heuristic approach for the NP-hard multi-pickup and delivery problem with time windows based on Ant Colony Optimization (ACO) algorithm. ACO is an algorithm that mimics the behaviour of ant colonies in efficiently finding the shortest path from the nest to a food source. The Smooth Max-Min Ant System algorithm is used as a rule to update pheromone in the ACO algorithm. Visual Studio 2019 is used as the program software. The test instances of 24 types are used in computational experiments. The experiments have been tested with 100, 200, 500, 1000, 2000, and 3000 iterations. The results are then compared with those generated by the CPLEX optimizer. Among three window types, the instances belonging to Normal Time Windows produced the best results. For the same node cases, the instances with long requests were solved faster than those with short requests due to having fewer requests that needed to work with than the latter. The results were assessed via a comparison with the Adaptive Large Neighborhood Search, CPLEX, in which the ACO algorithm performed well in most instances. Since mathematical programming could not handle instances with 400 nodes, it could be said that the problem is complicated and complex. This demonstrated the NP-hard characteristics of the multi-pickup and delivery problem with the time windows problem.**

***Index Terms*—Ant Colony Optimization; Multi-pickup and Delivery Problem; Time Windows; Smooth Max-Min Ant System Algorithm; NP-hard Problem; CPLEX Optimizer.**

I. INTRODUCTION

A multi-pickup and delivery problem with time windows (MPDPTW), which is introduced in [1], [2], is a variant of the Vehicle Routing Problem that is in the class of NP-hard [1]. According to [1], [2], each request in MPDPTW is satisfied by a vehicle picking up items from different locations to be shipped and unloaded at one common delivery location. Additionally, a time window (TW) is associated with each node. This time window allows scheduled pick-ups and deliveries concerning the node's start and end times. TWs are also indicated where the vehicles are contained to represent operation hours. The main objective is to minimize the route's overall costs, ensuring that required pick-ups and deliveries are satisfied.

In the MPDPTW, a mapping of one vehicle per request is made such that the single vehicle has to fulfil a single route made up of combined requests for pick-ups and deliveries. Moreover, vehicle tours have to be developed for precedence constraints (order in which nodes of a given request are visited) while reducing the overall routing cost. These constraints do not take precedence between the last visited pick-up and delivery nodes. The requirement is to fulfil pick-ups first and then visit the delivery node.

MPDPTW problem shares some characteristics with problems previously studied in the literature, namely the pick-up and delivery problem with time windows (PDPTW) and the sequential ordering problem (SOP). As the MPDTW is an extension of PDPTW and SOP, it is thus an NP-Hard problem [3]. The MPDPTW is one of the combinatorial optimization problems.

The Ant Colony Optimization proposed by [4] is a unique approach in which the algorithm simulates the behaviour of ant colonies, aiming to find the shortest path from the ant's nest to the food source based on the pheromone that the ants left on the way. This approach can be applied to many combinatorial optimization problems in the NP-hard class.

This paper applies an ACO algorithm based on smoothed max-min for ant system for MPDPTW. The main contributions of our paper are as follows:

- Presentation of a meta-heuristic approach (ACO with smoothed max-min for ant system algorithm) for the NP-hard Multi-pickup and Delivery Problem with Time Windows;
- Implementation of experiments with test instances of 24 types. For each type, five instances have been generated, resulting in a total of 120 instances in the tests;
- Providing a favourable comparison of results of our ACO method to ALNS and CPLEX.

In addition to this section, the rest of the paper includes (2) a literature review that provides a summary of approaches using previous works to solve MPDPTW, (3) Methodology that describes our methods of applying ACO algorithm with smooth max-min ant system for MPDPTW, (4) Results and Discussion, and (5) Conclusion.

II. LITERATURE REVIEW

The first work on MPDPTW was that of Naccache *et al.* (2018) [2], in which they used both exact and heuristic methods to solve the problem. While a branch-and-bound approach was applied with a new formulation to find solutions exactly, a hybrid adaptive large neighbourhood search algorithm with improvement operations was also proposed for the approximated solutions. The branch-and-cut algorithm was also applied for this problem in the paper of Aziez et al. (2020) [4], using several families of valid inequalities to strengthen the linear programming relaxations of the proposed formulations. As cited by [2] and [4], the MPDTW problem has similar characteristics with the sequential ordering problem and the pick-up and delivery problem with time windows. Therefore, the literature review of the two above problems is also considered. The following

briefly reviews the sequential ordering problem and the pick-up and delivery problem with time windows.

The sequential ordering problem (SOP) has been solved by several methods such as local search, branch-and-cut, and genetic. Local search algorithms were presented to address SOP problems using the concept of k-interchange by Savelsbergh (1985 [5], 1990 [6]). Ascheuer *et al.* (2001) constructed a new model to solve the SOP problems with the branch-and-cut algorithm [7]. A cooperative multi-thread parallelization strategy has been applied in the paper of Guerriero and Mancini (2003) [8]. The sequential algorithm that paralleled the heuristic rollout method was proposed for solving the SOP. Seo and Moon (2003) presented an improvement of the genetic algorithm to tackle the SOP [9]; the Voronoi quantized crossover adopted the complete graph representation was applied in their hybrid algorithm. Letchford *et al.* (2015) studied mathematical programming tools with two new multi-commodity flow formulations to address the SOP [10].

Similar to SOP, several approaches have been applied for the delivery problem with time windows (PDPTW). Pisinger and Ropke (2007) used a heuristic framework: the adaptive large neighbourhood search with robust algorithms to solve the PDPTW [11]. A parallel algorithm for PDPTW was proposed by Subramanian *et al.* (2010) [12]. The method was the integration of a multi-start heuristic containing a variable neighbourhood descent procedure and a random neighbourhood ordering in an iterated local search. The particle swarm optimization (PSO) was often used in works on the PDPTW. Ai and Kachitvichyanukul (2009) introduced a new formulation for the problem and a PSO algorithm that was performed using the method of decoding and representing a random key-based solution [13]. This year, Zhang et al. presented a modified version of the PSO algorithm for solving the PDPTW problem in which the sweep algorithm was applied in the decoding process to transform the particles to the matrices of vehicle allocation [14]. The PSO approach was continued to address the PDPTW in the paper of Goksal *et al.* (2013), in which a heuristic method combining PSO and neighbourhood descent algorithm applied for the local search was proposed [15].

Aside from these heuristic approaches, several exact methods were also applied for solving the PDPTW. Ropke *et al.* (2007) [16] built models for the problem based on two novel formulations and used the branch-and-cut algorithms to deal with the PDPTW. Ropke and Cordeau (2009) solved the problem based on a branch-and-cut-and price method. In their new proposed algorithm, they computed the lower bounds using the column generation approach in which the pricing subproblems consisting of the nonelementary and elementary shortest path problem were considered [17]. Baldacci *et al.* (2011) [18] introduced a novel algorithm for solving PDPTW using a formulation named set-partitioning–like integer. The branch-and-cut-and price algorithm was then applied in this paper to solve the issue of the integrality gap. Another exact algorithm for PDPTW was presented by Alyasiry *et al.* (2019) [19], in which the relaxed network flow model was constructed using fragments that were pick-up and delivery requests series which were started as well as ended by an empty vehicle.

Based on the above review, there are no doubt that only a few studies on methods applied for solving the MPDPTW problem. Hence, seeking other methods is still necessary.

Our paper focuses on using a meta-heuristic ACO algorithm with smoothed max-min for the ant system to deal with this MPDPTW problem.

III. METHODOLOGY

A. Problem Description

This section recalls the MPDPTW's description by Naccache et al. (2018) [15] as below:

There are $\boldsymbol{n}$ requests and $\boldsymbol{m}$ vehicles included in an MPDPTW's problem instance. $\mathbf{P = \{1, ..., p\}, D = \{p + 1, ..., p + n\}}$, and $\mathbf{R = \{r_1, ..., r_n\}}$ are the pick-up nodes set, the delivery nodes set, and the requests to be routed set, respectively, where $\mathbf{|D| = n}$ **and** $\boldsymbol{p \leq n}$ in which each set of pick-up nodes $P_r \subseteq$ P and delivery node $d_r \in$ D represents a request $r \in \mathbf{R}$. Each pick-up node belongs to exactly one set, and each request always contains at least one pick-up node. We also have customer nodes set N = P ∪ D, r(i) is the request associated with node $i \in N$ and K = {1, ..., m} is the available vehicles set.

The graph G = (V, A) contains the nodes V containing the customer nodes, starting node, and ending depot or delivery node. Each node i ∈ V has a service time s_i and a TW [a_i, b_i]. The parameters of the time window TW for each node corresponds to its start time a_i and end time b_i of operation.

The set of arcs is A = V × V minus arcs that lead to infeasible solutions: we omit arc (i, j) if the following occurs:

- i is a pick-up node and j is its delivery node if $bj < ai + si + tij$
- i is a delivery node and j is its pick-up node
- i is the start depot and j is a delivery node
- i is a pick-up node and j is the end of depot

A distance $dij \geq 0$ and a travel time $tij \geq 0$ are associated with each arc (i, j) ∈ A. Let A+ (i) and A− (i) be the sets of outgoing and incoming arcs from node i ∈ V.

The goal is to optimize route cost and provide requests for all vehicles, such as one request is only served by a vehicle. A solution for MPDPTW is to optimize route cost and divide requests for all vehicles such that one request is only served by a vehicle. An example of MPDPTW [15] is described in Figure 1.

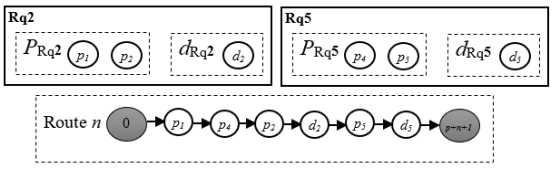

Figure 1. An MPDPTW example: Requests Rq2 and Rq5 inserted into route n

As shown in Figure 1, Rq2 and Rq5 are two requests; p_1 - p_4 are the pick-up nodes, d_1-d_6 are delivery nodes, and a route is associated with vehicle n. For instance, in the request in Rq2, to collect items to be delivered to d_2, p_1 and p_2 need to be visited. Similarly, for the request Rq5, p_4 and p_5 are necessary to be visited to collect items to be delivered to d_5. Note that p_2 is not directly visited after p_1 from the same request. In addition, the delivery node of request Rq2 is visited after all items have been collected from p_1 and p_2. The same applies to the request Rq5. Node p_4 is visited before node p_5 in the same request, and the delivery node d_5 is visited after all items have been collected from p_5 and p_4. Regarding precedence constraints, the two requests Rq2 and Rq5, are inserted into route n.

The mathematical formulation of MPDPT. The mathematical formulation of the MPDPTW problem uses the following decision variables:

- x_{ij}^k = 1 if arc (i, j) is traversed by vehicle k, 0 otherwise;
- y_{rk} = 1 if request r is visited by vehicle k, 0 otherwise;
- S_i indicates the beginning of service at node i ∈ V.
- The problem is then formulated as below:

$$\text{Min} \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \quad (1)$$

such that

$$\sum_{j \in A^+(i)} x_{ij}^k = y_{r(i)k} \qquad k \in K \;\; i \in N \quad (2)$$

$$\sum_{j \in A_{(i)}} x_{ji}^k = y_{r(i)k} \qquad k \in K \;\; i \in N \quad (3)$$

$$\sum_{j \in A^+(0)} x_{0j}^k \le 1 \qquad k \in K \quad (4)$$

$$\sum_{k \in K} y_{rk} = 1 \qquad r \in R \quad (5)$$

$$S_j \ge S_i + (s_i + t_{ij} + M) \sum_{k \in K} x_{ij}^k - M \qquad (i,j) \in A \quad (6)$$

$$a_i \le S_i \le b_i \qquad i \in V \quad (7)$$

$$S_{d_r} \ge S_i + s_i + t_{id_r} \qquad i \in P_r, r \in R \quad (8)$$

$$x_{ij}^k,\; y_{rk} \in \{0,1\} \qquad (i,j) \in A,\; r \in R,\; k \in K \quad (9)$$

$$S_i \ge 0 \qquad i \in V \quad (10)$$

Function (1) is the objective function that minimizes the overall transportation cost. The degree constraints (2) and (3) are in order to guarantee all nodes of a request belonging to the same vehicle. The constraint of at most K vehicles being used in the solution is ensured by (4), (5) makes sure that a request is to be served by only one vehicle. The schedule feasibility concerning TWs is guaranteed by constraints (6) and (7). Constraint (8) ensures the precedence order. The constraints of nature and the domain of the variables are imposed by (9) and (10). A big enough number M is equal to max $\{b_i + s_i + t_{ij} - a_j, 0\}$ for each constraint (7) [20].

B. Ant Colony Optimization

Ant colony optimization (ACO) and its most notable applications were proposed by [21]. Based on the name, it mimics how ant species demonstrate foraging by emitting pheromones. The path by which the pheromones are released creates a pattern that other colony members may follow. Ant colony optimization uses the same approach by finding a favourable path [21].

As shown in Table 1, four main factors decide the ACO algorithm's efficiency when it is applied for specified problems: building a construction graph and partial solution, determining heuristic information, and selecting update pheromones' rule.

Table 1
Algorithm of ACO Meta-Heuristic

Procedure ACO Meta-heuristic
Begin
Set parameters, initialize pheromone-marked paths
while termination condition not met do
Construct Ant Solutions
Apply Local Search (optional)
Update Pheromones
End while
Optimized solution
End;

Building a construction graph and partial solution depends on the problem's characteristics. The update pheromones' rule demonstrates the learning strategy of the algorithm. It is a common factor and is used to distinguish ACO algorithms. The rule, which is used in this report, is smoothed Max-Min Ant System.

C. Smoothed Max-Min Ant System

This Smoothed Max-Min Ant System (SMMAS) [22] is an improvement over the Max-Min Ant System algorithm [23]. This algorithm is simpler and more efficient than Max-Min Ant System. Its performance is tested experimentally through standard problems such as Traveling Salesman Problem and Production Scheduling Problem.

The SMMAS updates general pheromone for all trails instead of updating the pheromone trails of only the best ants. The rule that SMMAS follows will be shown in the next section.

D. ACO Algorithm using SMMAS for MPDPTW

In this section, MPDPTW is solved by an ACO algorithm based on L.P. Tran's work of ACO for vehicle routing problems [24], in which the SMMAS is applied for updating the pheromone step.

Construction graph. Given N is a set of customers (including pick-up nodes and delivery nodes) with N = n_1, n_2, . . ., n_n, M is a set of requests (including vehicle capacity, window times, combinations of picking-up before delivery) with M = m_1, m_2, . . ., m_n and E is a set of edges that connect nodes belonging to possible solutions, where a construction graph G = (N, E) for dealing with this combinatorial optimization problem by ACO is built, as shown in Figure 2.

The graph contains ***n*** levels in which V_o is the start and end node of each route. Each layer V_i includes customer nodes (both pick-up nodes and delivery nodes). Ants build a partial solution. Each ant that starts at V_o visits partially from layer 1 to layer n. At each layer, a random node will be selected to visit such that it would not break the constraints, and this selection relies on the probability that is calculated based on heuristic information and pheromone value following the SMMAS method.

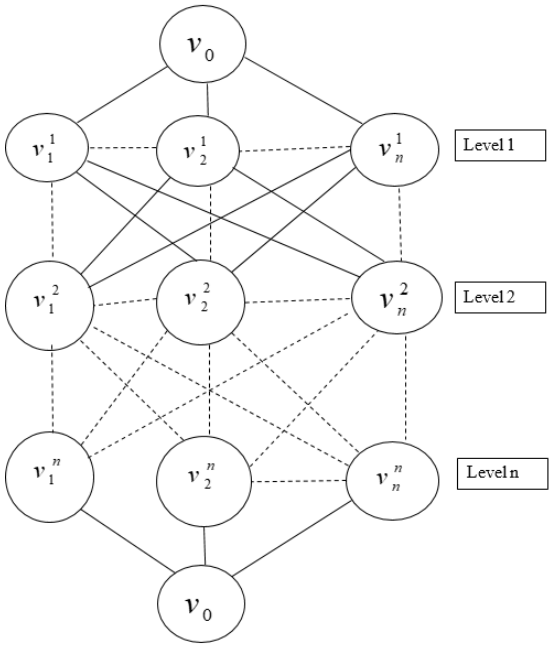

Figure 2. Construction Graph for MPDPTW

A solution for MPDPTW. A set of identified routes, and each route is served by a vehicle, is a solution for this problem. Each ant builds routes of the solution partially. Ants insert nodes into the current route such that these nodes would break the constraints. If the current solution cannot be inserted more nodes and is still unvisited, ants suspend this route and start a new one.

Let φ is a current unfinished solution executed by $r-1$ finished route and a route r^{th} in building solution. Suppose that i is the final node that is visited in the route r, Q_r is the capacity of the vehicle after visiting node i and U is a set of candidates that can visit after i. A node j is qualified if it has not been fulfilled and falls under one of the following conditions:

1. $0 < j \leq p$ and there exists a route that responds to all constraints in $U \cup \{p + i\}$.
2. $p < j \leq (p + n) \wedge j - p \in r$ here exists a route that responds to all constrains in $V\backslash\{j\}$.

These parameters prove that a possible route responds to all constraints after visiting node j. The problem is a Hamilton cycle.

j, which is inserted into the current solution φ, is randomly selected using the following equation:

$$P_{ij}^{\varphi} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}^{\varphi}]^{\beta}}{\sum_{d \in S_i^{\varphi}}[\tau_{ij}]^{\alpha}[\eta_{ij}^{\varphi}]^{\beta}} & j \in S_i^{\varphi} \\ 0 & j \notin S_i^{\varphi} \end{cases} \quad (11)$$

where P_{ij}^{φ} is the probability of selecting node j from the current node i. τ_{ij} identifies pheromone on the arc $(i\ j)$. S_i^{φ} is a set of qualified nodes that are visited after node i. Parameters α and β indicate the ratio of pheromone and heuristic information, respectively. η_{ij}^{φ} is heuristic information used to train the ants.

$$\eta_{ij}^{\varphi} = \frac{1}{H_1 + H_2} \quad (12)$$

where H_1, H_2 are heuristic information that is identified as follows:

$$H_1 = w_1 * d_{ij};\ w_1 \in [0,1] \quad (13)$$

H_1 is the information showing the distance between node i and node j, and w_1 is a parameter whose value is in [0,1]. In this work, the $w_1 = 0.1$.

$$H_2 = w_2 * (\text{departureTime} - bi);\ w_2 \in [0,1] \quad (14)$$

H_2 is the information showing the waiting time of ants to sever node j after visiting node i. departureTime is the total serving time up to the current time at node j. w_2 is a similar parameter as w_1, w_2 is set by 0.6.

A procedure of building solution is shown in Table 2.

Table 2
Procedure of MPDPTW Solution

Procedure MPDPTW_Solution
Begin
Input: U: set of qualified nodes, ItMax1: number of loops, i: current request, t: ending time of request i
Repeat
U' = U
Initialize a route r starting at node i and time t
Repeat
Select randomly a node d ∈ U' that respond constraints, remove d from U' and insert it into r at the most suitable position
If d is not inserted into r **then**
Go back to U'=U
End if
Until U' = ∅
Until current loop = ItMax1 or there exists a possible route responding to constraints in U
End;

Local search procedure. According to [21], generally, the best performing ACO algorithms make intensive use of the optional local search phase of the ACO meta-heuristic. Following this trend, the local search algorithm improves the solutions constructed by ants in this work. The ACO algorithm for MPDPTW is shown in Table 3.

Two heuristic algorithms, L1 and L2, are applied for MPDPTW, in which L1 relies on greedy strategy to reduce the travel distance for the identified route by exchanging the position of pick-up nodes in the same route randomly to derive the better route (if it is possible), L_2 is similarly built as L_1 except exchanging some requests of route i to any route j at the best position in the solution instead of exchanging pick-up nodes' position such that the total cost is minimum.

Table 3
ACO Algorithm for MPDPTW

Procedure MPDPTW_ACO
Begin
Initialize parameters, pheromone matrix, and m ants
Repeat
For k =1 to m **do**
Ants build a solution responding to all constraints
Repeat
Apply L_1 and L_2
Until the best solution cannot reduce the travel distance
Update the local pheromone
End for
Update the global pheromone
Until derive the best solution in the loop
Derive the best solution for the problem
End;

Test instances. This work uses the test instances that were used by [2]. These test instances were built upon existing PDPTW instances from [25] and available at [26].

Each instance type is characterized by a Time Windows (TWs) type, the maximum length of the requests and the number of nodes (instance size). An instance is defined depending on the window types as without (the TWs of the original node is deleted), with Normal (the TWs is slightly enlarged by opening it 150 units earlier and closing it 100 units later) or with Large TWs (the TWs is enlarged by opening it 300 units earlier and closing it 150 units later). For each instance, the minimum size of a request is two, as it includes one delivery point. Depending on the instance type, it can include at most 4 (Short requests) or 8 (Long requests) pick-up and delivery nodes. Finally, the instances contain 25, 50, 100 or 400 nodes. Five instances have been generated for each of the 24 instance types, resulting in a total of 120 instances in the tests [2].

A set of instances classified according to the characteristics is presented in Table 4**.**

Table 4
Test Instances [2]

	Without TWs		**Normal TWs**		**Large TWs**	
Instance size	Short requests	Long requests	Short requests	Long requests	Short requests	Long requests
25	W_4_25	W_8_25	N_4_25	N_8_25	L_4_25	L_8_25
50	W_4_50	W_8_50	N_4_50	N_8_50	L_4_50	L_8_50
100	W_4_100	W_8_100	N_4_100	N_8_100	L_4_100	L_8_100
400	W_4_400	W_8_400	N_4_400	N_8_400	L_4_400	L_8_400

For example, the instances of type l_8_400 represent Large TWs, long requests, 400 nodes. The format of each instance is as follows:

- The first row: The number of vehicles; The capacity of vehicle
- The second row: The depot (Starting and ending point of each route)
- Row 3^{rd} to n: Nodes
- Each row contains 8 parts corresponding to 8 columns:
  - Column 1: Node ID
  - Column 2: X
  - Column 3: Y
  - Column 4: Demand (Noted that the pick-up node 's demand < 0)
  - Column 5: Star TWs
  - Column 6: End TWs
  - Column 7: Node's status (depot: 0; pickup node: 0; delivery node: 1)
  - Column 8: The request ID

IV. Results and Discussion

The experiments have been performed on three computers, including a Core i7 4690 CPU, 32GB RAM, graphic card GTX1070 one (for Without TWs instances), a Xeon 1270p CPU, 32GB RAM, graphic card RTX3060Ti (for Normal TWs instances), and a Xeon E5-2673V3 CPU, 64GB RAM, graphic card GTX1070Ti (for Large TWs instances).

The parameters and their corresponding values are shown below:

- nAnt (Number of Ants): 100
- maxiteractions (Number of iterations): 2000
- rho (ρ: Evaporation rate): 0.03
- alpha and beta (α & β: the rates of pheromone and heuristic information): 1.0 and 2.0, respectively.

These parameter values are initialized to the values reported in [24]. The experiments have been tested with 100, 200, 500, 1000, 2000, 3000 iterations, as shown in Table 2. The running times rose significantly while the improvement of results did not change after reaching 2000 iterations. Hence, the option of 2000 iterations was selected.

Table 5
Iteration Testing

Instance	**Cost**	**Time (s)**	**Iteration**
N_4_100_1	14523.81	6.94	100
	14523.81	13.02	200
	14523.81	19.50	300
	14521.05	31.92	500
	14515.76	64.38	1000
	14515.76	122.07	2000
W_4_100_3	9901.19	11.44	100
	9650.39	22.13	200
	9656.66	49.99	500
	9622.24	98.36	1000
	9477.66	196.32	2000
	9477.66	299.89	3000
N_8_400_1	54242.75	514.88	1000
	53958.59	1021.84	2000
	53958.59	1486.39	3000
L_8_400_5	51470.27	913.59	1000
	51442.78	1824.17	2000
	51442.78	2553.83	3000

Table 5 illustrates the experiment's results of performing 24 instance types with the deviation being computed from the best solution value and the average solution. Due to the approximate characteristic of ACO, these results are computed from the average over each instance's replications. The ACO algorithm executes ten times over each instance.

Table 6
Experiment Results

Instance	# Requests	# Node	Best Solution	Average Solution	Time (ms)	Deviation (%)
W_4_25	7.8	26	2,541.45	3,079.90	13,665	**17.48%**
W_8_25	5	26.8	2,674.16	3,096.94	13,164	**13.65%**
W_4_50	17.2	50.6	4,450.24	5,123.91	43,658	**13.15%**
W_8_50	9.8	51	4,467.74	5,156.74	37,527	**13.36%**
W_4_100	33.8	101.2	8,182.58	8,683.25	165,441	**5.77%**
W_8_100	21.6	103.2	8,993.44	10,102.86	133,528	**10.98%**
W_4_400	132.4	400.8	28,387.84	30,064.51	883,492	**5.58%**
W_8_400	81.24	402.4	35,517.31	37,712.10	600,744	**5.82%**
N_4_25	7.8	26	4,186.68	4,734.83	12,296	**11.58%**
N_8_25	5	26.8	4,223.51	4,694.74	12,128	**10.04%**
N_4_50	16.2	50.6	7,763.31	8,928.98	37,971	**13.05%**
N_8_50	9.8	51	7,631.97	8,532.99	32,638	**10.56%**
N_4_100	34.2	156.6	14,270.24	15,391.04	119,898	**7.28%**
N_8_100	21.6	103.2	15,832.66	16,898.61	106,296	**6.31%**
N_4_400	133.8	401	45,632.10	49,274.44	1,199,176	**7.39%**
N_8_400	81.4	402.4	53,958.59	59,775.23	907,472	**9.73%**
L_4_25	7.48	24.8	3,101.33	3,977.96	12,772	**22.04%**
L_8_25	5	24	3,681.02	4,424.05	13,037	**16.80%**
L_4_50	10	47.8	6,221.15	7,500.54	35,179	**17.06%**
L_8_50	10.8	50.6	6,221.15	7,500.54	35,268	**17.06%**
L_4_100	34.2	101.2	11,179.40	12,224.99	131,668	**8.55%**
L_8_100	21.6	103.2	13,694.16	14,618.17	125,016	**6.32%**
L_4_400	133.8	401	38,483.84	40,877.26	600,976	**5.86%**
L_8_400	81.6	402.4	45,673.95	50,530.35	1,808,086	**9.61%**

Table 6 indicates that ACO performed well with 25, 50, 100 nodes than 400 nodes. The results of ACO in the former node cases were around those of ALNS. Among three window types, the instances belonging to Normal TWs got the best results. This suggested future work to find a better solution for a dataset of large TWs. For the same node cases, the instances with long requests were solved faster than those with short requests because the former had less numbers of requests that needed to work with than the latter. However, their cost is more expensive.

The analysis continues with a comparison of the ACO algorithm's results with Adaptive Large Neighborhood Search (ANLS) and CPLEX [2]. The results of this work will be assessed based on the ANLS and CPLEX output. A comparison of the ACO algorithm's results with ALNS is present in Table 7. The achieved results for all algorithms are the overall transportation cost (the objective function) tested on instances with a limit of 400 nodes.

As shown in Table 7, for best solution, ACO showed a better performance in most instances excepting W_4_400 and W_4_400 compared with ANLS and CPLEX while ANLS and CPLEX have the same results. However, mathematical programming (CPLEX) could not address instances with 400 nodes [2]. Similar to ANLS, ACO could handle the instances with 400 nodes in which ACO performed better in the Normal window type and the **Large** window type with long requests, whilst an opposite pattern was seen in the remained instances with 400 nodes.

V. CONCLUSION

This work presents a meta-heuristic solution to the NP-hard Multi-pickup and Delivery Problem with Time Windows (MPDPTW) based on the Ant Colony Optimization (ACO) algorithm in which an improvement of Max-Min Ant System (MMAS). This improved version is the Smoothed Max-Min Ant System (SMMAS) applied for the pheromone section. The performance of this ACO algorithm was tested on 24 instance types. Among three window types, the instances belonging to Normal Time Windows produced the best results. The results were assessed via a comparison with the ANLS, CPLEX wherein the ACO algorithm performed well in most instances. The MPDPTW problem is NP-hard due to the rapid increase in running time and complexity as the input data increases. This is proven by the inability of the solution (optimization) to handle 400 nodes.

Table 7
Comparison among algorithms' results

Instance	ACO	ALNS [15]	CPLEX [15]	Derivation to ANLS	Derivation to CPLEX
W_4_25	2,541.45	3,079.90	3,079.90	-17.48%	-17.48%
W_8_25	2,674.16	3,047.18	3,047.18	-12.24%	-12.24%
W_4_50	4,450.24	5,108.32	5,108.32	-12.88%	-12.88%
W_8_50	4,467.74	4,970.50	4,970.50	-10.11%	-10.11%
W_4_100	8,182.58	8,432.62	8,432.62	-2.97%	-2.97%
W_8_100	8,993.44	9,221.80	9,221.80	-2.48%	-2.48%
W_4_400	28,387.84	24,523.29	-	15.76%	-
W_8_400	35,517.31	29,462.98	-	20.55%	-
N_4_25	4,186.68	4,734.83	4,734.83	-11.58%	-11.58%
N_8_25	4,223.51	4,646.16	4,646.16	-9.10%	-9.10%
N_4_50	7,763.31	8,923.03	8,923.03	-13.00%	-13.00%
N_8_50	7,631.97	8,521.67	8,521.67	-10.44%	-10.44%
N_4_100	14,270.24	15,217.64	15,217.64	-6.23%	-6.23%
N_8_100	15,832.66	16,894.20	16,894.20	-6.28%	-6.28%
N_4_400	45,632.10	47,004.53	-	-2.92%	-
N_8_400	53,958.59	58,130.75	-	-7.18%	-
L_4_25	3,101.33	4,117.56	4,117.56	-24.68%	-24.68%
L_8_25	3,681.02	4,424.05	4,424.05	-16.80%	-16.80%
L_4_50	6,221.15	7,213.16	7,213.16	-13.75%	-13.75%
L_8_50	6,221.15	7,368.10	7,368.10	-15.57%	-15.57%
L_4_100	11,179.40	12,060.54	12,060.54	-7.31%	-7.31%
L_8_100	13,694.16	13,930.92	13,930.92	-1.70%	-1.70%
L_4_400	38,483.84	36,971.82	-	4.09%	-
L_8_400	45,673.95	46,406.75	-	-1.58%	-

REFERENCES

[1] C. Archettia, D. Feilletb, M. Gendreauc and M. G. Speranza, "Complexity of the VRP and SDVRP," *Transportation Research Part C: Emerging Technologies*, vol. 5, no. 19, pp. 741-750, 2011.

[2] S. Naccache, J. F. Côté and L. C. Coelho, "The multi-pickup and delivery problem with time windows," *European Journal of Operational Research*, vol. 1, no. 269, pp. 353-362, 2018.

[3] S. Naccache, J. F. Côté and L. C. Coelho, An Adaptive Large Neighborhood Search for the Multi-Pickup and Delivery Problem with Time Windows, CIRRELT, 2017.

[4] I. Aziez, J. F. Côté and L. C. Coelho, "Exact algorithms for the multi-pickup and delivery problem with time windows," *European Journal of Operational Research*, vol. 3, no. 284, pp. 906-919, 2020.

[5] M. W. Savelsbergh, "Local search in routing problems with time windows," *Annals of Operations research*, vol. 4, no. 1, pp. 285-305, 1985.

[6] M. W. Savelsbergh, "An efficient implementation of local search algorithms for constrained routing problems," *European Journal of Operational Research*, vol. 1, no. 47, pp. 75-85, 1990.

[7] N. Ascheuer, M. Fischetti and M. Grötschel, "Solving the asymmetric travelling salesman problem with time windows by branch-and-cut," *Mathematical programming*, vol. 90, no. 3, pp. 475-506, 2001.

[8] F. Guerriero and M. Mancini, "A cooperative parallel rollout algorithm for the sequential ordering problem," *Parallel Computing*, vol. 5, no. 29, pp. 663-677, 2003.

[9] D. I. Seo and B. R. Moon, "A hybrid genetic algorithm based on complete graph representation for the sequential ordering problem," in *Genetic and Evolutionary Computation Conference*, Seattle, WA, USA, 2004.

[10] A. N. Letchford and J. J. Salazar-González, "Stronger multi-commodity flow formulations of the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 3, no. 244, pp. 730-738, 2015.

[11] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Computers & operations research*, vol. 8, no. 34, pp. 2403-2435, 2007.

[12] A. Subramaniana, L. Drummonda and C. Bentesb, "A parallel heuristic for the vehicle routing problem with simultaneous pick-up and delivery," *Computers & Operations Research*, vol. 11, no. 37, pp. 1899-1911, 2010.

[13] T. J. Ai and V. Kachitvichyanukul, "A particleswarm optimization for the vehicle routing problemwith simultaneous pick-up and delivery," *Computers & Operations Research*, vol. 36, no. 5, pp. 1693-1702, 2009.

[14] I. Zhang, G. Sun, Y. Wu and F. Geng, "A modified particle swarm optimization for the vehicle routing problem with simultaneous pick-up and delivery," in *7th Asian Control Conference. IEEE*, 2009.

[15] F. P. Goksal, I. Karaoglan and F. Altiparmak, "A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pick-up and delivery," *Computers & industrial engineering*, vol. 1, no. 65, pp. 39-53, 2013.

[16] S. Ropke, J. F. Cordeau and G. Laporte, "Models and branch-and-cut algorithms for pick-up and delivery problems with time windows," *Networks: An International Journal*, vol. 4, no. 49, pp. 258-272, 2007.

[17] S. Ropke and J. F. Cordeau, "Branch and cut and price for the pick-up and delivery problem with time windows," *Transportation Science*, vol. 3, no. 43, pp. 267-286, 2009.

[18] R. Baldacci, E. Bartolini and A. Mingozzi, "An exact algorithm for the pick-up and delivery problem with time windows," *Operations research*, vol. 2, no. 59, pp. 414-426, 2011.

[19] A. M. Alyasiry, M. Forbes and M. Bulmer, "An exact algorithm for the pick-up and delivery problem with time windows and last-in-first-out loading," *Transportation Science*, vol. 6, no. 53, pp. 1695-1705, 2019.

[20] G. Desaulniers, O. Madsen and S. Ropke, "Chapter 5: The vehicle routing problem with time windows," in *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, Society for Industrial and Applied Mathematics, 2014, pp. 119-159.

[21] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine 1*, vol. 4, no. 1, pp. 28-39, 2006.

[22] D. Do, "Ant colony optimization and its application," Ph.D. Dissertation, DHQGHN, 2012.

[23] T. Stützle and H. H. Hoos, "MAX–MIN ant system," *Future generation computer systems*, vol. 16, no. 8, pp. 889-914, 2000.

[24] T. Tran, "Ant colony optimization for vehicle routing problems," Master thesis, DHQGHN, 2019.

[25] H. Li and A. Lim, "A metaheuristic for the pick-up and delivery problem with time windows," *International Journal on Artificial Intelligence Tools*, vol. 02, no. 12, pp. 173-186, 2003.

[26] H. Li and A. Lim, "Li Lim benmark," [Online]. Available: https://www.sintef.no/projectweb/ top/pdptw/li-lim-benchmark/Li and Andrew Lim.. [Accessed 10 November 2020].