8-2023

An adaptive large neighborhood search for heterogeneous vehicle routing problem with time windows

Minh Pham Kien NGUYEN

Aldy GUNAWAN
Singapore Management University, aldygunawan@smu.edu.sg

Vincent F. YU

Mustafa MISIR

Citation

NGUYEN, Minh Pham Kien; GUNAWAN, Aldy; YU, Vincent F.; and MISIR, Mustafa. An adaptive large neighborhood search for heterogeneous vehicle routing problem with time windows. (2023). *Proceedings of the 19th IEEE International Conference on Automation Science and Engineering (CASE), Auckland, New Zealand, 2023 Aug 26-30*. 2023,.
Available at: https://ink.library.smu.edu.sg/sis_research/8316

An Adaptive Large Neighborhood Search for Heterogeneous Vehicle Routing Problem with Time Windows

Minh Pham Kien Nguyen[1], Aldy Gunawan[2], Vincent F. Yu[1], Mustafa Mısır[3]

2023 IEEE 19th International Conference on Automation Science and Engineering (CASE) | 979-8-3503-2069-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/CASE56687.2023.10260380

Abstract— **The heterogeneous vehicle routing problem with time windows (HVRPTW) employs various vehicles with different capacities to serve upcoming pickup and delivery orders. We introduce a HVRPTW variant for reflecting the practical needs of crowd-shipping by considering the mass-rapid-transit stations, as the additional terminal points. A mixed integer linear programming model is formulated. An Adaptive Large Neighborhood Search based meta-heuristic is also developed by utilizing a basic probabilistic selection strategy, i.e. roulette wheel, and Simulated Annealing. The proposed approach is empirically evaluated on a new set of benchmark instances. The computational results revealed that ALNS shows its clear advantage on the instances with the increasing number of vehicles, especially compared to commercial software, CPLEX.**

I. Introduction

The development of public infrastructure has brought a improved and convenient life to people around the globe. The effects of the pandemic COVID-19 have also significantly boosted the required demands through e-commerce platforms in delivering small-scale items over short periods of delivery time rather than large-scale goods with a long delivery time. As a result, a multi-type vehicle has been adopted for achieving this objective where each one has a unique configuration. This is known as the Heterogeneous Vehicle Routing Problem (HVRP), as an extension of the VRP [1]. The HVRP adopts more than one single type of vehicle. Recently, advanced technology on robotics with a significant number of inventions on different types of vehicles such as drones and electric vehicles, have prompted many studies to integrate them into the HVRP [2], [3]. Sitek and Wikarek [4] consider parcel lockers and delivery options.

The present study introduces the HVRP with time windows (HVRPTW), considering the public transportation configurations, such as mass rapid transit (MRT) stations, as additional terminals. They handle the delivery tasks using an outsourced team from the terminals, which requires a significant large demand that small-capacity vehicles cannot carry. The configuration of the heterogeneous vehicle fleets involves a number of small-capacity and large-capacity vehicles with different operating costs per unit time. The actual operating time of vehicles is also taken into account, including the transportation and service time at each node.

[1]Minh Pham Kien Nguyen & Vincent F. Yu are with National Taiwan University of Science and Technology, 43 Keelung Road, 106335, Taipei, Taiwan, `d10901807@gapps.ntust.edu.tw`, `vincent@mail.ntust.edu.tw`

[2]Aldy Gunawan is with Singapore Management University, 81 Stamford Road, 178902, Singapore, `aldygunawan@smu.edu.sg`

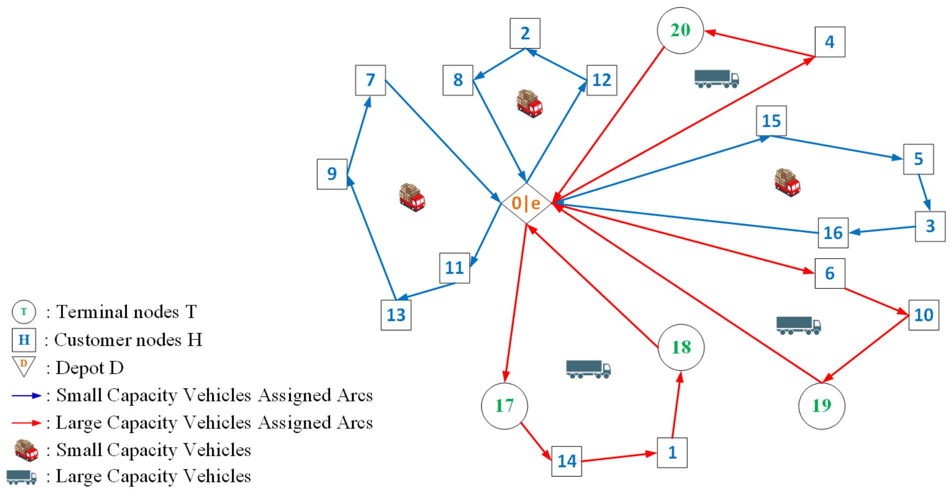
[3]Mustafa Misir with Division of Natural and Applied Sciences, Duke Kunshan University, Duke Avenue No: 8, 215316, Kunshan, Jiangsu, China

The main contribution of this work is the adoption of multiple types of vehicles in delivering goods to metro terminals as transshipment locations with significantly large service time for later delivery to the customers. A new set of benchmark instances is also generated. An Adaptive Large Neighborhood Search (ALNS) metaheuristic is devised as a practical algorithm. ALNS [5] reconstructs the new solutions from an initially generated solution using a set of destroy and repair operators. Those operators are managed and selected using the Roulette Wheel selection procedure. We adopt Simulated Annealing (SA) that accepts worsening solutions referencing a probability value. The results show that the proposed approach can tackle HVRPTW, achieving good-quality solutions within shorter computational times, compared against CPLEX.

II. Problem Description

The HVRPTW is defined as a directed graph $G = (N, A)$ with a node set N and an arc set A, as illustrated in Fig. 1. The node set $N = \{0, e\} \cup H \cup T$ includes the depot node as the start node and its virtual duplicate node $\{0, e\}$ as the end node, H is a set of nodes that represents customers who require the home delivery, and T is the set of terminal stations such as mass rapid transit (MRT) stations, where parcel lockers are located to consolidate orders. These orders may be sent to the other stations so that the customers can collect their orders at those stations. The delivery operation from terminals to other stations is assumed to be outsourced. Thus, this activity is discarded in our model.

Each customer $i \in H$ has a demand $d_i > 0$ and wants the order to be delivered within time window $[l_i, u_i]$. Each terminal $o \in T$ has a significantly large demand d_o due to order consolidation with the same time window that equals the operation time of the depot $[l_o, u_o] \equiv [\mathbf{l_0}, \mathbf{u_0}]$. For delivering orders, a set of vehicles V is divided into two categories or subsets: (1) a number of delivery vehicles $v^1 \in V$ with small capacity $\mathbf{Q}^{v^1}$ that only serve customers requiring home delivery $i \in H$, and (2) a number of large-capacity $\mathbf{Q}^{v^2}$ vehicles $v^2 \in V$ that can fulfill demands from both terminals $o \in T$ and customers requiring home delivery $i \in H$. To simplify the mathematical model, we introduce a general capacity parameters $\mathbf{Q}^v = \mathbf{Q}^{v^1} \cup \mathbf{Q}^{v^2}$. Each vehicle $v \in V$ carries a load of lv_i^v from node $i \in N$ at time $t_i^v \geq l_i \geq \mathbf{l_0} \geq 0$, where t_i^v indicates the vehicle v arrival time at node i, to fulfill the demands d_j at node $j \in N : j \neq i$ with travel time t_{ij} and arrive at node j at time $t_j^v \geq t_i^v + t_{ij} + \Delta_j^v : l_j \leq t_j^v \leq u_j$ with the load of $lv_j^v = lv_i^v - d_j \geq 0$. Here, $\Delta_j^v = \max(0, l_j - (t_i^v + t_{ij}))$ is the idle (or waiting) time



2
20
4
7
8
12
15
5
9
0|e
3
16
11
6
10
13
18
19
17
14
1
T : Terminal nodes T
H : Customer nodes H
D : Depot D
: Small Capacity Vehicles Assigned Arcs
: Large Capacity Vehicles Assigned Arcs
: Small Capacity Vehicles
: Large Capacity Vehicles

Fig. 1: An Illustration of the HVRPTW

of vehicle $v \in V$. Each node $j \in N$, except the depot, has a service time of $s_j > 0$, Each vehicle $v \in V$ will deliver the order at this node at $t_j^v = t_i^v + t_{ij} + \Delta_j^v + s_j$. Each vehicle $v \in V$ will be compensated for its delivery activities with a cost per unit of time of C^v respectively. We introduce a set of tuples $\langle i,j,v\rangle \in P$ addressing the pairs between location and vehicles that are designed to address the small and large vehicles respectively. Let X_{ij}^v be the binary decision variable for the assignments of vehicle $v \in V$ from nodes i to j $\forall (i,j) \in A : i \neq j, i \neq \{e\}, j \neq 0$. In more details, if $X_{ij}^v = 1$, then vehicle $v \in V$ will serve location $j \in N : j \neq 0$ from location $i \in N : i \neq j, i \neq \{e\}$ and 0 otherwise. The objective is to find the minimum total compensation for all the vehicles. Let $N^* = N \setminus \{0, e\}$. The complete mathematical model is shown below.

$$min \sum_{\substack{\langle i,j,v\rangle \in P \\ v \in V}} C^v \times (t_{ij} + s_j) \times X_{ij}^v \qquad (1)$$

Subject to:

$$\sum_{\langle i,j,v\rangle \in P} X_{ij}^v = 1 \qquad \forall j \in N^* \qquad (2)$$

$$X_{0j}^v \le 1 \qquad \forall \langle 0,j,v\rangle \in P, \forall v \in V \qquad (3)$$

$$\sum_{\langle i,e,v\rangle \in P} X_{ie}^v \le 1 \qquad \forall v \in V \qquad (4)$$

$$\sum_{\langle 0,j,v\rangle \in P} X_{0j}^v - \sum_{\langle i,e,v\rangle \in P} X_{ie}^v = 0 \qquad \forall v \in V \qquad (5)$$

$$\sum_{\langle i,j,v\rangle \in P} X_{ij}^v - \sum_{\langle j,k,v\rangle \in P} X_{jk}^v = 0 \qquad \forall v \in V, \forall j \in N^* \qquad (6)$$

$$\sum_{\substack{\langle i,o,v^1\rangle \in P \\ o \in T}} X_{io}^{v^1} = 0 \qquad \forall v^1 \in V \qquad (7)$$

$$\sum_{\langle i,o,v\rangle \in P} X_{io}^v = 1 \qquad \forall o \in T \qquad (8)$$

$$lv_i^v - lv_j^v - \mathbf{Q}^v \times (1 - X_{ij}^v) \le d_j \qquad \forall \langle i,j,v\rangle \in P, \forall j \in H \qquad (9)$$

$$lv_i^v - lv_j^v + \mathbf{Q}^v \times (1 - X_{ij}^v) \ge d_j \qquad \forall \langle i,j,v\rangle \in P, \forall j \in H \qquad (10)$$

$$lv_i^{v^2} - lv_o^{v^2} - \mathbf{Q}^{v^2} \times (1 - X_{io}^{v^2}) \le d_o \qquad \forall \langle i,o,v^2\rangle \in P, \forall o \in T \qquad (11)$$

$$lv_i^{v^2} - lv_o^{v^2} + \mathbf{Q}^{v^2} \times (1 - X_{io}^{v^2}) \ge d_o \qquad \forall \langle i,o,v^2\rangle \in P, \forall o \in T \qquad (12)$$

$$lv_i^v \le \mathbf{Q}^v \times \sum_{\langle i,j,v\rangle \in P} X_{ij}^v \qquad \forall v \in V, \forall i \in N^* \qquad (13)$$

$$lv_0^v \le \mathbf{Q}^v \times \sum_{\langle 0,j,v\rangle \in P} X_{0j}^v \qquad \forall v \in V, \forall j \in N^* \qquad (14)$$

$$t_j^v - t_i^v - s_j - \Delta_j^v - (\mathbf{u_0} - \mathbf{l_0} + 1) \times (1 - X_{ij}^v) \le t_{ij} \qquad \forall \langle i,j,v\rangle \in P, \forall j \in H \qquad (15)$$

$$t_j^v - t_i^v - s_j - \Delta_j^v + (\mathbf{u_0} - \mathbf{l_0} + 1) \times (1 - X_{ij}^v) \ge t_{ij} \qquad \forall \langle i,j,v\rangle \in P, \forall j \in H \qquad (16)$$

$$\Delta_o^v = 0 \qquad \forall o \in T \cup e, \forall v \in V \qquad (17)$$

$$\Delta^v = \sum_{i \in N} \Delta_i^v \qquad \forall v \in V \qquad (18)$$

$$t_i^v + t_{ij} + \Delta_j^v + (\mathbf{u_0} - \mathbf{l_0} + 1) \times (1 - X_{ij}^v) \ge l_j \qquad \forall \langle i,j,v\rangle \in P, \forall j \in H \qquad (19)$$

$$t_i^v + t_{ij} + \Delta_j^v \le u_j + (\mathbf{u_0} - \mathbf{l_0} + 1) \times (1 - X_{ij}^v) \qquad \forall \langle i,j,v\rangle \in P, \forall j \in H \qquad (20)$$

$$t_i^v + t_{io} + \Delta_o^v \le u_o + (\mathbf{u_0} - \mathbf{l_0} + 1) \times (1 - X_{io}^v) \qquad \forall \langle i,o,v\rangle \in P, \forall o \in T \cup e \qquad (21)$$

$$l_j + (\mathbf{u_0} - \mathbf{l_0} + 1) \times (X_{ij}^v - 1) \le t_j^v \qquad \forall \langle i,j,v\rangle \in P, \forall j \in H \qquad (22)$$

$$u_j + (\mathbf{u_0} - \mathbf{l_0} + 1) \times (1 - X_{ij}^v) \ge t_j^v \qquad \forall \langle i,j,v\rangle \in P, \forall j \in H \qquad (23)$$

$$u_o + (\mathbf{u_0} - \mathbf{l_0} + 1) \times (1 - X_{io}^{v^2}) \ge t_o^{v^2} \qquad \forall \langle i,o,v^2\rangle \in P, \forall o \in T \cup e \qquad (24)$$

$$t_i^v \le (\mathbf{u_0} - \mathbf{l_0} + 1) \times \sum_{\langle i,j,v\rangle \in P} X_{ij}^v \qquad \forall v \in V, i \in N \qquad (25)$$

$$t_0^v = 0 \qquad \forall v \in V \qquad (26)$$

$$X_{ij}^v \in \{0,1\} \qquad \forall \langle i,j,v\rangle \in P \qquad (27)$$

$$t_i^v \geq 0, lv_i^v \geq 0, \Delta_i^v \geq 0 \qquad \forall v \in V, \forall i \in N \quad (28)$$

$$\Delta^v \geq 0 \qquad \forall v \in V \quad (29)$$

The objective function (1) minimizes the total compensation for all deliveries. Constraint (2) ensures that all customers will be served. Constraints (3) - (4) ensure that each vehicle will depart and return to the depot after delivery. The numbers of inbound and outbound vehicles from the depot have to be the same (Constraint (5)). Constraint (6) ensures that vehicles will leave after visiting or servicing nodes. Constraint (7) prohibits small vehicles approaching terminal nodes and Constraint (8) guarantees the visit at terminal points. Constraints (9) - (10) cover the sub-tour elimination constraints. Constraints (11) - (12) ensure that large-capacity vehicles will not deliver more or less than the terminals' demands. Constraints (13) - (14) guarantee that vehicles will not violate their capacities.

Constraints (15) and (16) ensure accuracy in vehicles' operating time. Constraint (17) allows large-capacity vehicles to visit terminal nodes at any time, and Constraint (18) computes the total waiting time of each vehicle. The vehicles' arrival time at the assigned nodes is bounded by the nodes' time windows (Constraints (19) - (20)). Constraint (21) ensures that vehicles will not return to the depot or approach terminal nodes later than the nodes' upper bound of the time windows. Constraints (22) - (24) have the same function as (19) - (21). Constraint (25) ensures that vehicles' time will only be accounted at the node they depart, and constraint (26) is initially set to time = 0 as the beginning of the delivery operation. The natural range of decision variables is given by Constraints (27) - (29).

III. Proposed ALNS

We explain the framework of ALNS in Algorithm 1. Algorithms 2, 3, and 4, which are parts of the entire framework, will be explained as well. First, an initial solution X^{Init} is randomly generated by assigning nodes to vehicles randomly with respect to the time windows and capacity constraints, which later be assigned to the global solution X^{Global} and the local solution X^{Local} (Algorithm 1: Lines 2-3).

The variable `BestCost` is assigned with the objective value $F(X^{Global})$, the non-optimal counter ξ is set to 0, while the iterations counter υ for updating mechanism proportions is set to 1 (Algorithm 1: Lines 4-5). The global loop is initiated with maximum iterations Υ by resetting the local loop counter ϕ to 1 with the maximum allowed iterations for the local loop = Φ (Algorithm 1: Lines 6-8). The local loop then generates a number of removal nodes N^{Exc} to be excluded from the current solution X^{Init}. Destroy operator D will be selected from the set of destroy operators M_D by generating a random number $Rand(0,1)$ that is higher than the proportion of the prior operator π_υ^{D-1} and less than the current operator proportion π_υ^D (Algorithm 1: Lines 9-10). The current solution X^{Init} then removes N^{Des}, which can be higher than the generated destroyed nodes N^{Exc} using the drawn destroy operator D. A repair operator R will be selected from the repair set M_R using a random number $Rand(0,1)$ that is bounded by the proportion of the selected operator π_υ^R and its prior candidate π_υ^{R-1} to construct a new solution X^{Init} with the excluded nodes N^{Des} from the destroyed ones (Algorithm 1: Lines 11-13). This work adopts the destroy and repair operators [6].

Algorithm 1: The Proposed ALNS

```
Input:
    Set of destroy operator $M_D$, repair operator $M_R$, and update mechanism $\mathtt{U}$; Normalization mechanism $K$; The set of selection probabilities for destroy operators $\pi_0^D$ and repair operators $\pi_0^R$; The score set of destroy operators $\delta_0^D$ and repair operators $\delta_0^R$ for update mechanism's decision; Maximum available iterations for generating local solutions $\Phi$; Maximum available iterations for updating the destroy and repair operators $\Upsilon$; Maximum non-global improvement iterations $\Xi$; Objective Function $F(X)$.
Output:
    Global solution $X^{Global}$; Total Compensation $F(X^{Global})$; Total Travel Time $Travel$
1  Initialize:
2      Generate an Initial Solution $X^{Init}$
3      $X^{Global} \leftarrow X^{Init}; X^{best} \leftarrow X^{Init}$
4      $\mathtt{BestCost} \leftarrow F(X^{Global})$
5      $\upsilon \leftarrow 1; \xi \leftarrow 0$
6      while $\upsilon \leq \Upsilon$ do
7          $\phi \leftarrow 1$
8          while $\phi \leq \Phi$ do
9              Generate a number of remove nodes
                 $N^{Exc} \leftarrow Rand(1, H+T)$
10             $D(N^{Exc}, \pi_\upsilon^D) \xleftarrow{Rand(0,1)} M_D : \pi_\upsilon^{D-1} \leq$
                 Rand(0,1) $< \pi_\upsilon^D$
11             $X^{Init} \xleftarrow{D} X^{Init} - N^{Des} : N^{Des} \in [N^{Exc}, H+T]$
12             $R(N^{Exc}, \pi^D) \xleftarrow{Rand(0,1)} M_R : \pi_\upsilon^{R-1} \leq$
                 Rand(0,1) $< \pi_\upsilon^R$
13             $X^{Init} \xleftarrow{R} X^{Init} + N^{Exc}$
14             $\mathtt{U}(X^{Global}, X^{best}, X^{Init})$
15             $\delta_\phi^D \xleftarrow{\mathtt{U}(X^{Global}, X^{best}, X^{Init})} \delta_{\phi-1}^D + \delta_D$
16             $\delta_\phi^R \xleftarrow{\mathtt{U}(X^{Global}, X^{best}, X^{Init})} \delta_{\phi-1}^R + \delta_R$
17         if $F(X^{Global}) < \mathtt{BestCost}$ then
18             $\mathtt{BestCost} \leftarrow F(X^{Global})$
19             $\xi \leftarrow 0$
20         else
21             $\xi \leftarrow \xi + 1$
22         $\pi_\upsilon^D \xleftarrow{K(\delta_\Phi^D)} \pi_{\upsilon-1}^D; \pi_\upsilon^R \xleftarrow{K(\delta_\Phi^R)} \pi_{\upsilon-1}^R$
23         if $\xi = \Xi$ then
24             return $X^{Global}, F(X^{Global}), Travel$
25 return Output
```

The first four operators represent the destroy operators:
Random Removal: this operator randomly selects a node in the initial solution X^{Init} using the Roulette Wheel procedure and inserts it into the Removal List N^{Des}.
Route Removal: this operator randomly chooses a route in the current solution, excludes all covered nodes in that route from the solution, and inputs them into the Removal List.
Zone Removal: this operator randomly generates a region with four pairs of coordinates (x_{low}, y_{low}); (x_{low}, y_{high}); (x_{high}, y_{low}); and (x_{high}, y_{high}). Then, all nodes belonging to the generated region, except the depot, are excluded from the solution and are added into the Removal List.
Worst Removal: it removes N^{Des} nodes from the current solution that consumes large travel times among routes in the descending order and adds them into the Removal List.

We also implement the following repair operators [6]:
Random Best Recover: it re-inserts the excluded nodes from

N^{Des} by selecting nodes using the Roulette Wheel procedure and then searching for its best insert location within the existing route or a new generated feasible route.

Random Best with Noise Recover: this operator considers a noise score Δ_s by selecting the maximum value among the multiplication results of a random generated factor $Rand(-1,1)$ with a small fixed parameter $\pi = 0.1$ to the current route's maximum distance with the service time between the covered nodes. This noise will be added to the re-insert costs of the excluded nodes, and these nodes are then re-inserted based on their best locations so far.

Greedy Recover: this operator sorts the re-insert costs of all excluded nodes in the ascending order and adds them to their best feasible locations. **Greedy with Noise Recover**: this is similar to the second one. The re-insert cost will be added by a noise score Δ_s mentioned before and then performs the same procedure as **Greedy Recover**. **The Farthest Best Recover**: this operator adds the nodes that are further from the depot first, which is opposite to the **Closest Best Recover**, which re-inserts the closer nodes. After the sequence of nodes is created, the operator will insert the nodes based on the best positions.

The Updated mechanism $\texttt{U}$ (Algorithm 2) will be initiated, which results in a pair of destroy and repair operator scores, δ_D and δ_R, respectively. These scores are used to update the total score so far of the respective destroy operator δ_ϕ^D and repair operator δ_ϕ^R at local iteration ϕ (Algorithm 1: Lines 14-16). After the local loop ϕ reaches its maximum allowed counters Φ, the objective value $F(X^{Global})$ will be compared with the $\texttt{BestCost}$ to check if a new global optimum was found. If any updates with the optimum solutions happen, then the non-optimal counter ξ is reset to 0 or increased by 1 otherwise. The proportion sets of both Destroy π_υ^D and Repair operators π_υ^R are then updated by normalizing the set of operators' achieved score so far, $K(\delta_\phi^D)$ and $K(\delta_\phi^R)$ (Algorithm 1: Lines 17-22). Two conditions to terminate Algorithm 1 are: (1) the number of non-upgrade iterations ξ for the global score to reach its maximum counter Ξ or (2) the global iteration υ reaches its maximum value Υ.

The proposed ALNS updates the existing solution by implementing the Updated Mechanism. First, the current solution is checked if it is currently in the solution pool or not before examining the pair conditions: the policy on the number of available vehicles and the obtained objective values, $F(X^{Global})$ and $F(X^{best})$. If it is a new solution, then a set of designed scenarios with respective actions is implemented (Algorithm 2: Lines 2-27):

Achieve a new global solution $(F(X^{Global}) > F(X^{Init}))$: the new solution X^{Init} will be assigned to both global solution X^{Global} and local solution X^{best}. The global solution pool χ^{Global} will be cleared to add the current solution in addition to the historical solution pool χ. Its cost $F(X^{Init})$ is added to the Cost pool C and the non-local update counter λ will be reset to 0. This achievement will set the reward scores for the applied destroy and repair operators $\texttt{Reward}^D$ and $\texttt{Reward}^R$ to the Best Achievement scenario with $\Delta_D(1)$ and $\Delta_R(1)$ (Algorithm 2: Lines 3-7).

Discover a permutation of the current global solution $(F(X^{Global}) = F(X^{Init}))$: the global solution pool χ^{Global} will not be cleared while the remaining activities are kept similar to the first scenario (Algorithm 2: Lines 8-12).

Obtain a new local solution $(F(X^{best}) > F(X^{Init}))$: the update procedure involves the local solution X^{best} only. In addition, χ^{Global} will not add the found solution while the solution pool χ and the Cost pool C will still record the new solution and its cost. The non-local update counter λ is reset to 0, and the reward scores for the selected destroy and repair operators $\texttt{Reward}^D$ and $\texttt{Reward}^R$ are set to the Medium Achievement scores $\Delta_D(3)$ and $\Delta_R(3)$, respectively (Algorithm 2: Lines 13-16).

Attain a new permutation of the current local solution $(F(X^{best}) = F(X^{Init}))$: this will check if the non-local optimal updates counter λ exceeds its maximum allowance Λ or not. The true condition will trigger the Reconstruct mechanism that assigns a recorded global solution $X^{Global} \in \chi^{Global}$ to both current local optimal solution X^{best} and current solution X^{Init} and reset the non-local update counter λ to 0 (Algorithm 2: Lines 18-19).

Obtain a worse-than-local-optimal-solution $(F(X^{Init} < F(X^{best}))$: this triggers the Simulated Annealing algorithm (Algorithm 3) to see whether the current solution is accepted as a new local optimal with the respective reward scores for the selected destroy and repair operators $\texttt{Reward}^D$ and $\texttt{Reward}^R$, the updated solution pool χ, and the non-local update counter λ (Algorithm 2: Lines 24-25).

The initial solution is otherwise accepted and assigned to the local optimal $X^{best} \leftarrow X^{Init}$. The solution pool χ will record the current solution X^{Init}, and the solution's cost $F(X^{Init})$ is also added to its respective pool while the non-local update counter is increased by 1. The reward scores for the respective applied operators $\texttt{Reward}^D$ and $\texttt{Reward}^R$ are set to the Worse Achievement score $\Delta_D(2)$ and $\Delta_R(2)$, respectively (Algorithm 2: Lines 20-23).

If the number of available vehicles is invalid, then the Reconstruct Solution mechanism will be triggered and the non-local update counter will be added by 1 (Algorithm 2: Lines 26-27). If the initial generated solution is a member of the solution pool $X^{Init} \in \chi$, a random generator will be initiated with a chance of 50% that the Reconstruct Solution mechanism is triggered, and the remaining 50% will force the initial solution to revert to the local solution $X^{Init} \leftarrow X^{best}$. In this case, the Worst Reward score set is assigned to the destroy and repair operators' $\texttt{Reward}^D \leftarrow \Delta_D(4)$ and $\texttt{Reward}^R \leftarrow \Delta_R(4)$ in addition to the incremental of 1 for the non-local update counters (Algorithm 2: Lines 28-33).

A brief description of Simulated Annealing is explained in Algorithm 3 with the input values of $\alpha = 0.9$, $\epsilon = 10^{-3}$, and $(L_p; U_p) = (1, 50)$ [7]. A checking procedure is done to see if the current global iteration υ is less than 3 or not. If true, then the initial temperature T_0 is set to the absolute difference between the iteration's maximum recorded cost so far $Max(C(\upsilon,:))$ and its minimum recorded cost $Min(C(\upsilon,:))$(Algorithm 3: Line 2-3); otherwise, we adopt the procedure for generating the initial temperature. For more

Algorithm 2: Updated Mechanism

```
Input:
    Global solution $X^{Global}$; Local solution $X^{best}$; Initial solution
    $X^{Init}$; Set of rewards for Destroy and Repair Operators $\Delta_D$ and $\Delta_R$;
    Set of objective values obtained $C$; Set of solutions obtained so far
    $\chi$; Set of global solutions so far $\chi^{Global}$; Number of available small
    vehicles $V^1$; Number of available large vehicles $V^2$; Maximum Number
    of Iterations without Improvement $\Lambda$; Objective Function $F(X)$.
Output:
    Updated initial solution $X^{Init}$; Updated global solution $X^{Global}$;
    Updated local solution $X^{best}$; Reward for destroy/repair operator
    Reward$^D$/Reward$^R$; Set of Costs $C$, Set of Solutions $\chi$, Set of
    Global Solutions $\chi^{Global}$
1  Initialize:
2      if $X^{Init} \notin \chi$ then
3          if $v^2 < V^2$ and $v^1 < V^1 : v^1$ and $v^2 \in X^{Init}$ and
             $F(X^{Global}) > F(X^{Init})$ then
4              $(X^{Global}, X^{best}) \leftarrow X^{Init}$
5              $\chi^{Global} \leftarrow \varnothing$; $\chi^{Global} \leftarrow \chi^{Global} + X^{Global}$;
                $\chi \leftarrow \chi + X^{Init}$
6              $C \leftarrow C + F(X^{Init})$; $\lambda \leftarrow 0$;
7              Reward$^D \leftarrow \Delta_D(1)$; Reward$^R \leftarrow \Delta_R(1)$
8          else if $v^2 < V^2$ and $v^1 < V^1 : v^1$ and $v^2 \in X^{Init}$ and
             $F(X^{Global}) = F(X^{Init})$ then
9              $(X^{Global}, X^{best}) \leftarrow X^{Init}$
10             $\chi^{Global} \leftarrow \chi^{Global} + X^{Global}$; $\chi \leftarrow \chi + X^{Init}$
11             $C \leftarrow C + F(X^{Init})$; $\lambda \leftarrow 0$
12             Reward$^D \leftarrow \Delta_D(1)$; Reward$^R \leftarrow \Delta_R(1)$
13         else if $v^2 < V^2$ and $v^1 < V^1 : v^1$ and $v^2 \in X^{Init}$ and
             $F(X^{best}) > F(X^{Init})$ then
14             $X^{best} \leftarrow X^{Init}$; $\chi \leftarrow \chi + X^{Init}$
15             $C \leftarrow C + F(X^{Init})$; $\lambda \leftarrow 0$
16             Reward$^D \leftarrow \Delta_D(3)$; Reward$^R \leftarrow \Delta_R(3)$
17         else if $v^2 < V^2$ and $v^1 < V^1 : v^1$ and $v^2 \in X^{Init}$ and
             $F(X^{best}) = F(X^{Init})$ then
18             if $\lambda > \Lambda$ then
19                 Reconstruct Solution; $\lambda \leftarrow 0$
20             else
21                 $X^{best} \leftarrow X^{Init}$; $\chi \leftarrow \chi + X^{Init}$
22                 $C \leftarrow C + F(X^{Init})$; $\lambda \leftarrow \lambda + 1$
23             Reward$^D \leftarrow \Delta_D(2)$; Reward$^R \leftarrow \Delta_R(2)$
24         else if $v^2 < V^2$ and $v^1 < V^1 : v^1$ and $v^2 \in X^{Init}$ then
25             $\chi$, $(X^{Init}, X^{Best}$, Reward$^D$; Reward$^R$; $\lambda) \leftarrow$
               Simulated Annealing
26         else if Other Cases then
27             Reconstruct Solution; $\lambda \leftarrow \lambda + 1$
28     else
29         if $Rand(0,1) > 0.5$ then
30             Reconstruct Solution
31         else
32             $X^{best} \leftarrow X^{Init}$
33         $\lambda \leftarrow \lambda + 1$; Reward$^D \leftarrow \Delta_D(4)$; Reward$^R \leftarrow \Delta_R(4)$
34 return Output
```

Algorithm 3: The Proposed ALNS: Simulated Annealing Process [7]

```
Input:
    Initial Solution $X^{Init}$; Local Solution $X^{best}$; Set of objective value
    obtained so far $C(v, \Phi)$; Non-improvement in local loop value $\lambda$;
    Acceptance ratio $\alpha \geq 0.8$; Step ratio $\varepsilon \leq 10^{-3}$; Range $(L_p; U_p)$ for
    generating real number $p \geq 1$.
Output:
    Initial Solution $X^{Init}$; Local Solution $X^{best}$; Reward for destroy/re-
    pair operator Reward$^D$/Reward$^R$; Non-improvements in local loop
    $\lambda$
1  Initialize:
2      if $v < 3$ then
3          $T_0 \leftarrow Max(C(v,:)) - Min(C(v,:))$
4      else
5          $p \leftarrow Rand(L_p; U_p)$
6          $i \leftarrow 1$; $\sigma \leftarrow 0$
7          while $i \leq v$ do
8              $\sigma \leftarrow \sigma + Max(C(i,:)) - Min(C(i,:))$
9          $T_0 \leftarrow \frac{-\sigma}{v \times log(\alpha)}$
10         $i \leftarrow 1$; $\eta \leftarrow 0$; $\zeta \leftarrow 0$
11         while $i \leq v$ OR $|E[T_0] - \alpha| > \varepsilon$ do
12             $\eta \leftarrow e^{\left(\frac{-Max(C(i,:))}{T_0}\right)}$
13             $\zeta \leftarrow e^{\left(\frac{-Min(C(i,:))}{T_0}\right)}$
14             $E[T_0] \leftarrow \frac{\eta}{\zeta}$
15             $T_0 \leftarrow T_0 \times \left(\frac{log\left(E\left[T_0\right]\right)}{log\left(\alpha\right)}\right)^{\frac{1}{p}}$
16     if $T_0 \leq 0$ then
17         $T_0 = Rand(0,1) \times Min(C(v,:))$
18     if $Rand(0,1) < e^{\left(\frac{F(X^{best}) - F(X^{Init})}{T_0}\right)}$ then
19         $X^{best} \leftarrow X^{Init}$
20         $\lambda \leftarrow 0$
21         Reward$^D \leftarrow \Delta_D(2)$; Reward$^R \leftarrow \Delta_R(2)$
22     else
23         $X^{Init} \leftarrow X^{best}$
24         $\lambda \leftarrow \lambda + 1$
25         Reward$^D \leftarrow \Delta_D(4)$; Reward$^R \leftarrow \Delta_R(4)$
26 return Output
```

details, please refer to [7]. The Reconstruct Mechanism plays a significant role in completing our algorithm by re-assigning the initial solution X^{Init} and the local solution X^{best} to a historical recorded global solution in its respective set $X^{Global} \in \chi^{Global}$ when an invalid solution is crafted from the destroy and repair operators in Algorithm 4.

IV. Computational Results

New benchmark instances are introduced by modifying the well known Gehring & Homberger's 200 customer benchmark instances c1_2_1 to c2_2_10. The instances with the indices of c2 (c2_2_1 to c2_2_10) have more number of small and large vehicles. Data are available upon request. The modified attributes include the vehicles' capacities, quantities, and operation costs, the number of nodes for home and terminal deliveries, the terminal nodes' capacity, the service time, and the time windows of respective nodes.

Experiments were executed on a computer with Intel Core i7-12700 CPU @ 2.10 GHz processor, 16.0 GB RAM (Windows 10 Education 64-bit) and solved by CPLEX 12.10.0.0, limited to three hours of computational times.The results are compared with the ones of our ALNS using Visual Studio Code with C++ platform version 1.75.0. For the ALNS, each instance is solved 5 times with the pre-defined input parameters: Global Iterations = 500, Local Iterations = 1000, Best Achievement Score = 100, Medium Achievement Score = 25 and Worse Achievement Score = 50.

Two types of experiments are conducted with the former is done with the Destroy and Repair operators, which are selected through the Roulette Wheel procedure (Algorithms 2 and 3), while the latter selects operators randomly. Computational results show that both Roulette Wheel and random selection procedures perform well on solving the problem. It outperforms CPLEX's non-optimal outcomes on 14 out of 20 instances. The ALNS with Roulette Wheel's outcomes'

Algorithm 4: The Proposed ALNS: Reconstruct Solution

```
Input:
    Initial Solution $X^{Init}$; Local Solution $X^{best}$;
    Global Solution $X^{Global}$; Set of global solutions so far $\chi^{Global}$.
Output:
    Initial Solution $X^{Init}$; Local Solution $X^{best}$.
1 Initialize:
2     if $Size(\chi^{Global} > 1)$ then
3         $X^{Init} \leftarrow X^{Global} : X^{Global} \equiv Rand(\chi^{Global})$
4         $X^{best} \leftarrow X^{Global}$
5     else
6         $X^{Init} \leftarrow X^{Global}$
7         $X^{best} \leftarrow X^{Global}$
8 return Output
```

TABLE I: Experiments Results

Instances	CPLEX	RunTime in sec.	Roulette Wheel Procedure				Random Procedure			
			Avg.	Best	Gap (%)	Avg. Time	Avg.	Best	Gap (%)	Avg. Time
c1_2_1	**4486***	348	4524	4524	0.85	9.7	4531.6	4524	1.0	7.1
c1_2_2	**5646***	836	5720	5720	1.31	12.4	5732.8	5720	1.54	11.3
c1_2_3	**10452***	1789	10520.8	10520	0.66	17	10526.4	10520	0.71	14.9
c1_2_4	**10180***	3884	10242	10242	0.61	20.9	10242	10242	0.61	17.9
c1_2_5	**16260**	10800	16344	16344	0.52	56.7	16352	16344	0.57	28.4
c1_2_6	5370	10800	5314.4	5308	−1.04	12.8	5306.4	**5302**	−1.18	10.2
c1_2_7	7870	10800	7843.2	7802	−0.34	18.2	7850.4	**7798**	−0.25	15.7
c1_2_8	**11842**	10800	11968	11938	1.06	28.8	11968	11938	1.06	19.9
c1_2_9	11662	10800	11652	11652	−0.09	21.9	11648	**11642**	−0.12	18
c1_2_10	22798	10800	22770	**22770**	−0.12	77	22770	**22770**	−0.12	53.3
c2_2_1	9178	10800	8538.4	**8490**	−6.97	24	8534	8514	−7.02	18.9
c2_2_2	9682	10800	9649.2	**9596**	−0.34	26.9	9704	9648	0.23	17.2
c2_2_3	16894	10800	16886.8	**16810**	−0.04	30.2	16924.8	16876	0.18	26.1
c2_2_4	21946	10800	21548	**21494**	−1.81	41.8	21554.8	21522	−1.78	33.2
c2_2_5	–	10800	29803.2	**29776**	–	60.6	29813.2	29784	–	41.3
c2_2_6	7648	10800	7460.4	**7356**	−2.45	14.9	7492	7382	−2.04	14.5
c2_2_7	10508	10800	9965.6	**9866**	−5.16	17.5	10018.4	9992	−4.66	17.4
c2_2_8	15726	10800	15090.4	**14986**	−4.04	38.8	15112.8	15012	−3.9	29.2
c2_2_9	19672	10800	18682	**18664**	−5.03	51.1	19063.2	18966	−3.09	42.4
c2_2_10	–	10800	34393.2	**34274**	–	82.7	34428	34370	–	72.4

*: CPLEX's Optimal Results

average costs are lower than the Random ones, especially on the second group of instances that are marked as $c2\_2\_*$, except for instances $c1\_2\_6$, $c1\_2\_9$, and $c2\_2\_1$.

The former approach also offers better stability than the latter one when making comparisons between both methods' outcomes' variances and their results' gaps with the CPLEX's best-known-so-far are also lower than the results provided by the random procedure. The Roulette Wheel procedure's coefficient of variances are consistently below 0.5% for all instances while for the random one, some instances have the coefficient variances more than 1%. We can conclude that our ALNS can handle the problem with a limited resources compared to CPLEX with shorter computation times especially for the second group of instances.

We also analyze the effects of removing a pair of destroy and repair operators for the Roulette Wheel approach by considering four scenarios in solving five randomly selected instances: without Route Removal and Greedy Recover operators (Scenario 1), without Route Removal and Greedy with Noise Recover operators (Scenario 2), without Worst Removal and Greedy Recover operators (Scenario 3), and without Worst Removal and Greedy with Noise Recover operators (Scenario 4), as shown in Table II.

Most of the scenarios' results are worse compared to the original Roulette Wheel and Random procedure results, except for the instances c2_2_9, which has a slightly better average result than the one of the Random procedure when being implemented with Roulette Wheel procedure (Scenario 1). The Greedy with Noise can help improving the results better than the original Greedy operators while the Worst removal is worse than the Route removal when applying the same recover operator themselves, except for instance c2_2_8 with the removal of Greedy with Noise recover operator.

TABLE II: Operators Evaluation

Instances	c1_2_6	c1_2_8	c2_2_7	c2_2_8	c2_2_9
CPLEX	5370	11842	10508	15726	19672
RW	5314.4	11968	9965.6	15090.4	18682
Random	5306.4	11968	10018.4	15112.8	19063.2
Scenario 1	5329.2	11980.4	10350.8	15594	19036.4
Scenario 2	5334.4	11983.6	10387.2	15628.8	19248.4
Scenario 3	5332.8	11995.2	10381.6	15611.6	19084
Scenario 4	5334.4	12012.4	10407.6	15665.6	19138

V. Conclusion

We introduce a heterogeneous vehicle routing problem with time windows. It is applicable for delivery activities in e-commerce platforms where the demands are small-scale items that are needed to be delivered at some convenient pickup points with a large number of requested orders. A mixed integer programming model is formulated and ALNS is also proposed. It has shown the effective in handling the newly developed benchmark instances with shorter computational times, limited resources.

The limitation of this study is a lack of appropriate input parameters. Future studies can focus on more sophisticated algorithms that can improve the solution quality, such as adopting hyper-heuristics for diversifying the solution or finding suitable input parameters.

References

[1] G. D. Konstantakopoulos, S. P. Gayialis, and E. P. Kechagias, "Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification," *Operational Research*, pp. 1–30, 2020.

[2] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5816–5823, 2021.

[3] X. Bai, W. Yan, and S. S. Ge, "Distributed task assignment for multiple robots under limited communication range," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, pp. 4259–4271, 2021.

[4] P. Sitek and J. Wikarek, "Capacitated vehicle routing problem with pick-up and alternative delivery (cvrppad): model and implementation using hybrid approach," *Annals of Operations Research*, vol. 273, no. 1-2, pp. 257–277, 2019.

[5] E. Demir, T. Bektaş, and G. Laporte, "An adaptive large neighborhood search heuristic for the pollution-routing problem," *European journal of Operational Research*, vol. 223, no. 2, pp. 346–359, 2012.

[6] C. Friedrich and R. Elbert, "Adaptive large neighborhood search for vehicle routing problems with transshipment facilities arising in city logistics," *Computers & Operations Research*, vol. 137, p. 105491, 2022.

[7] W. Ben-Ameur, "Computing the initial temperature of simulated annealing," *Computational Optimization and Applications*, vol. 29, pp. 369–385, 2004.