
# The Vehicle Routing Problem with Pickup and Delivery, Time Windows and Priority

带取送，时间窗和优先级约束的车辆路径问题


| | |
|---|---|
| Name | Ruimeng Chang |
| ID | 1716293 |
| Date | 05/05/2021 |
| Supervisor | Min Wen |

# Abstract

The Vehicle Routing Problem with Pickup and Delivery and Time Windows (VRPPDTW) aims to find a set of routes to complete transportation requests within specified time ranges. This work introduced one problem based on the VRPPDTW where membership status is assigned to each request. For non-members' requests, violation of time constraints is allowed at the expense of objective penalty. The problem is named as the VRPPDTW with Priority. One linear programming model was presented with the objective of minimizing a weighted sum of travel distance and service delay.

An Adaptive Large Neighborhood Search algorithm was introduced and the modified operator was discussed. The algorithm exhibited good performance for our model when tested on small instances. The proposed method was applied on 99 instances adapted from the VRPPDTW benchmark instances to compare the results obtained from applying different objective weights in the model. According to the experiments, if we increase the weight imposed on distance in the model, distance would be shorter while delay would be larger in the solution. The speed of delay increase is larger than that of distance decrease. For any given instance, it is possible to achieve the desired balance between cost and customer satisfaction by adjusting the objective weights. The flexibility of our model increases when the problem size increases or the member percentage decreases in the data.



**Keywords:** vehicle routing problem with pickup and delivery and time windows, adaptive large neighborhood search



带取送和时间窗约束的车辆路径问题(VRPPDTW) 旨在找到合适的路径在给定的时间内完成运输请求。本文基于此问题延申，为运输请求关联优先级。该问题中请求分为两类，会员等级的请求与非会员等级的请求，同时时间窗限制只对会员等级的请求有效。该问题被命名为带取送，时间窗和优先级约束的车辆路径问题，以最小化距离成本和延误加权和为目标函数的线性规划模型在文中给出。

本文介绍并分析了基于自适应大领域搜索的算法，并具体讨论了算法中算子的修改。该方法在应用于小型数据集求解本文模型时表现良好。该方法被应用在99个

基于VRPPDTW基准实例生成的数据实例中，以此来比较模型不同权重比产生的结果。实验表明当模型中距离成本的权重增大时，结果呈现距离减小而延误增加的趋势，同时延误增加的速度较快。对于任意给定的数据，成本与顾客满意度的平衡可以通过调节模型系数来达到。该模型的灵活度随问题规模的增大而增大，随会员等级的请求数量减少而增大。

关键词： 带取送和时间窗约束的车辆路径问题，自适应大领域搜索

---

# 1 Introduction

Transportation industry is the central concern of city logistics and has a huge impact on economy, society and environment (Cattaruzza et al., 2017). Common examples in daily life include parcel delivery, material collection, goods storage, waste collection, home delivery services and electrical appliance reparation services (Barceló et al., 2017). Urban transportation system aims to minimize the cost while meeting customers' requirements, which is a complex problem, especially on a large scale. New challenges emerge as a result of customers' increasingly diverse and demanding requests along with the constantly changing traffic information. Furthermore, policies and regulations impose additional difficulties on the problem. As businesses involving transportation put more and more emphasis on providing responsive service while reducing the cost at the same time, the importance of routing problems cannot be underestimated (Toth and Vigo, 2002).

Among all kinds of problems arising in city transportation, those request goods to be transported from and to specified location is of particular interest due to their commonality. Bus routing and scheduling, cargo transportation by in-port ships, parcel delivery and door-to-door food delivery are typical examples. Normally, the pickup service and delivery service are required to be finished within a certain range of time specified by the customer, otherwise customers would be unsatisfied about the service quality. This problem is called the vehicle routing problem with pickup and delivery and time windows (VRPPDTW).

In addition to the classical VRP which seeks to find routes that visit all given locations with a fixed fleet of vehicles subject to the objective of minimizing the travel distance and travel time, the VRPPDTW requires additional constraints ensuring that pickup service is completed before delivery service with the same vehicle within a time slot. Some research was based on applications of the VRPPDTW in logistics, including dial-a-ride problem (Wilson et al., 1976) and courier services (Shen et al., 1995).

Considering the case of food delivery, some customers pay monthly fees to attain membership for better service quality. These members would assume the service they request be completed within the time range according to their willingness. Although the service provider seeks to minimize the total delay, non-members may still experience delay of service if resources are limited. The first type of time constraints is named as hard time window and the later one as soft time window. We name this problem as the VRPPDTW with Priority and it is highly adaptive to other scenarios when customers are hierarchic with different levels of demand for service quality. However, this problem receives little attention and is worth looking into.

In this work, we studied the VRPPDTW with Priority in detail and proposed a linear programming model. An Adaptive Large Neighborhood Search (ALNS) approach based on Ropke and Pisinger (2006) was selected and introduced for this problem. Experiments of different objective weights were performed on instances with different member percentages generated based on Li and Lim (2003). The behavior of our solution is sensitive to the change of objective weights and the influence would be amplified by problem size and the presence of non-member requests.

The remaining of the article is organized as follows. In Section 2, related works of the VRPPDTW in the literature are briefly introduced. In Section 3, a formal description of our problem with all the constraints and assumptions is given. In Section 4, notations and mathematical models used in this problem are detailed. In Section 5, metaheuristic selected for this problem is discussed. In Section 6, computational experiments are presented and discussed. In Section 7, limitations are discussed. This work is concluded in Section 8.

# 2 Literature Review

The Vehicle Routing Problem (VRP) was first introduced by Dantzig and Ramser (1959) and has attracted an increasing number of scholars in operations research over the years. Many variants for the VRP were introduced for application-oriented reasons, among which the most important ones related to this work are the Vehicle Routing Problem with Time Windows (VRPTW) and the Vehicle Routing Problem with Pickup and Delivery (VRPPD).

The VRPTW was first proposed by Solomon (1987) and a set of heuristics were discussed in this article. This variant has additional time window constraints requiring the start time when the vehicle starts its service falls within a certain time slot.

The VRPPD was discussed first by Min (1989). It imposes additional coupling constraints and precedence constraints where a pair of mixed services including both pickup and delivery is requested to be completed within one route and pickup service must be finished before the start of delivery service. In the literature, different types of the VRPPD have been studied. Based on the type of demand and route structure, these works can be divided into three types (Toth and Vigo, 2014):

(i) One-to-Many-to-One problem, where requests are required to be transported either from one depot to multiple locations or from multiple locations to one depot. For example, companies that want to deliver their new products to customers from the warehouse and recycle used products at the same time may be modeled as one such problem. Related literature in this area includes Min (1989); Montané and Galvão (2002).

(ii) Many-to-Many problem, where goods picked up from any pickup location can be delivered to any delivery location. One example is delivering homogeneous commodities to customers from multiple warehouses. Hernández-Pérez and Salazar-González (2004); Psaraftis (1980) conducted research on problems of this type.

(iii) One-to-One problem, where each request is paired with specified pickup and delivery location. Door-to-door parcel delivery is one example as different requests are associ-

ated with different combinations of starting points and destinations. Related literature in this area includes Lin (1965); Nanry and Barnes (2000); Li and Lim (2003).

The VRPPDTW combines the characteristics of the VRPTW and the VRPPD and our VRPPDTW with Priority is one extension associating customer requests with different priority levels. Related literature for this problem is scarce.

According to Braekers et al. (2016), different variants of the VRP have received growing attention between 2009 and 2015, including the VRPPDTW. Different methods have been proposed in the literature to solve the VRPPDTW. Generally, most of these methods can be categorized as one of the followings: exact algorithms, heuristics and metaheuristics.

One exact algorithm was proposed by Sexton and Bodin (1985) as approximate algorithms based on Benders decomposition. Dumas et al. (1991) presented a Set-Partitioning formulation of the problem and proposed a Column Generation approach to solve the pricing subproblems. Instances with up to 55 requests were tested and solved efficiently. Several attempts to solve this variant with Dynamic Programming were made in Psaraftis (1980); Desrosiers et al. (1986). Ruland and Rodin (1997) formulated and explored the polyhedral structure of the VRPPD and applied a Branch-and-Cut algorithm solving problems of size up to 15. In Lu and Dessouky (2004), a two-index multi-depot version of the problem was formulated and valid inequalities for the LP-relaxation were discussed. In Ropke and Cordeau (2009), new classes of inequalities were introduced and two different pricing subproblems were tested. The approach of the latter one remains to be the best with respect to the size of the instances that can be solved optimally (Baldacci et al., 2011).

To solve large-scale problems, heuristic and metaheuristic methods were extensively explored because of the inefficiency of exact algorithms (Braekers et al., 2016). Earlier works used Iterative Local Search (ILS) with neighborhoods generated by k-interchange operator (Psaraftis, 1983; Van der Bruggen et al., 1993), which was proposed first by Lin (1965) and proved efficient for the VRP with one vehicle. Solomon (1987) described several tour construction heuristics including Savings Heuristics and Insertion Heuristics.

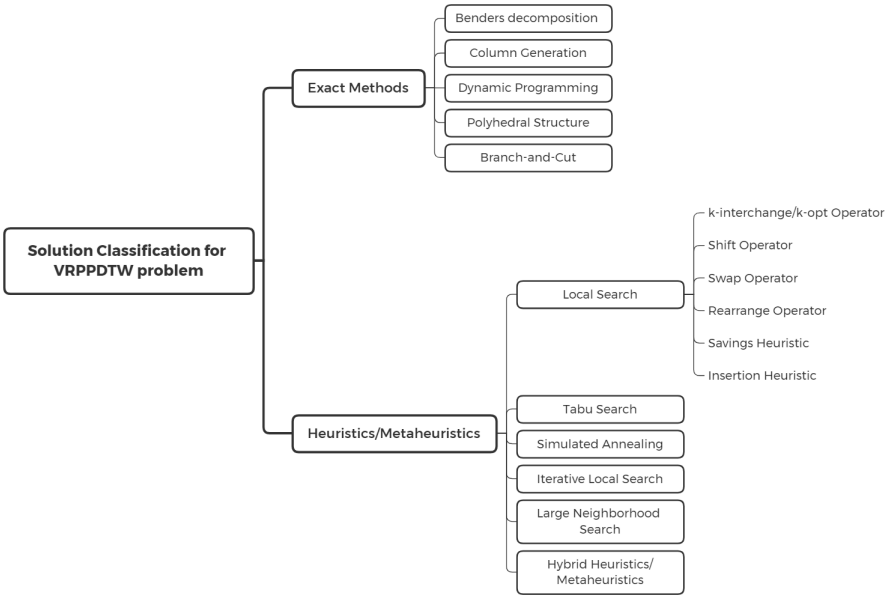



Figure 1: Solution Method Classification for the VRPPDTW

The first metaheuristic for the VRPPDTW was introduced by Nanry and Barnes (2000). A Reactive Tabu Search with the local search neighborhood generated by the following operations: (i) shift requests between routes; (ii) swap requests between routes; (iii) relocate requests within route; were applied to solve small instances up to 50 requests. A Tabu-embedded Simulated Annealing approach was introduced by Li and Lim (2003) using similar local search procedure as presented above. Another contribution of this research was proposing new test instances by adapting the VRPTW benchmark instances proposed by Solomon (1987), which later became the standard test instances for the VRPPDTW. Gendreau et al. (2006) proposed an Ejection Chain Neighborhood (ECN) and applied an Adaptive memory-based Tabu Search process on a dynamic vehicle routing problem. Several methods based on the Large Neighborhood Search (LNS) were presented in the literature. Bent and Van Hentenryck (2006) introduced a two-stage hybrid algorithm where a Simulated Annealing approach was implemented to minimize the number of vehicles in the

first stage while the LNS was applied to minimize the travel distance in the second. Ropke and Pisinger (2006) introduced an Adaptive Large Neighborhood Search (ALNS) approach where two stages were performed based on the same objective function with different dynamic adjusting parameters. This algorithm achieved remarkable efficiency by improving more than half of the benchmark test instances.

# 3 Problem Description

In this problem, we consider a fleet of vehicles and assign each of them to meet the requests of customers along its route. A request made by one customer is composed of a pair of services including pickup service from assigned location and delivery service to the customer. Each request is associated with the transportation of commodity with positive weight from pickup to delivery location. In addition, requests should first be picked up and then delivered by the same vehicle.

Each vehicle has a limited capacity, requiring the load after visiting each location not exceeding the maximum capacity. For the problem to have a feasible solution, the weight of each commodity is assumed to be smaller than the maximum vehicle capacity.

For each location, an associated demand value represents the weight of commodity to be transported. For each pickup location, the demand is defined to be positive and equal to commodity weight, so that the vehicle load would increase the amount of the commodity weight after visiting pickup location. For each delivery location, the demand is correspondingly defined to be the opposite number of the demand of the associated pickup location. For depots, demands are defined as 0.

Service time is defined to be the minimum time required between arrival and departure at each location. It models the time for loading and unloading. The service time is defined as 0 at both depots as visiting depots does not consume time.

To model the time requirements, each pickup and delivery location would specify its time window, the earliest and latest acceptable time within which service shall be started.

Since customers would not experience unqualified service for early arrival, vehicles are permitted to wait until the earliest service time to start service upon early arrival without any loss. Taking the hierarchy of customers into consideration, customers are divided into two groups, those with membership status and those not. Requests made by members are classified with higher priority. Hence, a route that provides service later than the latest allowable time at locations associated with members' requests is defined as infeasible, while violation of time window constraints is allowed for locations associated with non-members' requests at the cost of penalty in the objective. All vehicles are dispatched from the start depot at time 0 and are required to finish service at the end depot within certain time range. We define the time range as the time window for the depots to impose restriction on the maximum routing time. Depots are set with membership status in order to have hard time window restriction.

With constraints stated above, the aim of this problem is to find a feasible set of routes that both minimize the cost and maximize the degree of customer satisfaction. Two goals are introduced: (1) minimize the total routing distance; (2) minimize the time violation penalty. The objective in our model is to minimize a weighted sum of routing distance and time violation penalty.

To further illustrate the model, one example with 5 pairs of requests $R_i = (P_i, D_i)$ is presented, where $P_i$ and $D_i$ are the associated pickup and delivery locations respectively. Information related to different locations is shown in Table 1. The locations and associated membership statuses are displayed in Figure 2. Depots $E$ and $S$ coincide and $S$ is marked to represent both depots. In this example, only requests $R_1 = (P_1, D_1)$ and $R_3 = (P_3, D_3)$ are made by non-members, allowing time window constraints violation. Locations are given as Cartesian coordinates and Euclidean distance is used as distance between two locations. Only one vehicle with capacity 90 is available to service the customers.

The instance was tested on two sets of parameters : (a) $\alpha = 3, \beta = 1$; (b) $\alpha = 1, \beta = 3$; where $\alpha$ and $\beta$ are weights of the total travel distance and the total delay in the objective function respectively. The solutions and objective function values are given in Table 2 and

Table 1: Information of each location for example problem

| Request | Node | Time Window | Demand | Service Time | Location | Membership |
|---|---|---|---|---|---|---|
| | $S$ | [0,1150] | 0 | 0 | (45,55) | Y |
| $R_1$ | $P_1$ | [31,226] | 10 | 90 | (52,72) | N |
| | $D_1$ | [313,545] | -10 | 90 | (72,52) | N |
| $R_2$ | $P_2$ | [138,320] | 30 | 90 | (45,70) | Y |
| | $D_2$ | [358,528] | -30 | 90 | (60,75) | Y |
| $R_3$ | $P_3$ | [337,595] | 10 | 90 | (62,69) | N |
| | $D_3$ | [569,819] | -10 | 90 | (60,55) | N |
| $R_4$ | $P_4$ | [317,550] | 30 | 90 | (65,72) | Y |
| | $D_4$ | [561,721] | -30 | 90 | (60,66) | Y |
| $R_5$ | $P_5$ | [25,105] | 10 | 90 | (48,65) | Y |
| | $D_5$ | [800,960] | -10 | 90 | (68,45) | Y |
| | $E$ | [0,1150] | 0 | 0 | (45,55) | Y |

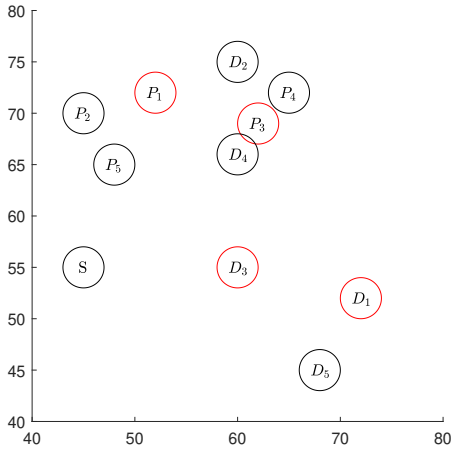
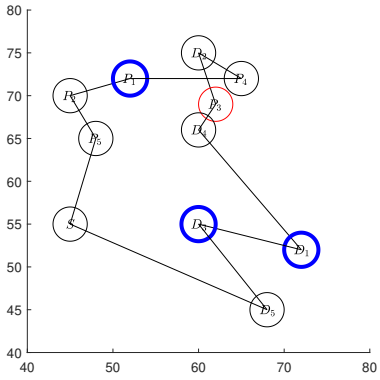
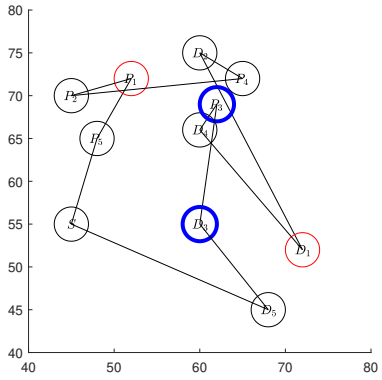



Figure 2: Initial setting for the example problem. Locations associated with requests by members are colored by black while non-members by red.



(a) $\alpha = 3, \beta = 1$



(b) $\alpha = 1, \beta = 3$

Figure 3: Solution route for example problem. Nodes which violate time window constraints are colored in blue.

Table 2: Solution route for example problem

| Node | Time Window | Start Time | Load | Delay |
|---|---|---|---|---|
| $S$ | [0,1150] | 0.00 | 0 | 0 |
| $P_5$ | [25,105] | 25.00 | 10 | 0 |
| $P_2$ | [138,320] | 138.00 | 40 | 0 |
| $P_1$ | [31,226] | 235.28 | 50 | 9.28 |
| $P_4$ | [317,550] | 338.28 | 80 | 0 |
| $D_2$ | [358,528] | 434.11 | 50 | 0 |
| $P_3$ | [337,595] | 530.44 | 60 | 0 |
| $D_4$ | [561,721] | 624.04 | 30 | 0 |
| $D_1$ | [313,545] | 732.48 | 20 | 187.48 |
| $D_3$ | [569,819] | 834.85 | 10 | 15.85 |
| $D_5$ | [800,960] | 937.66 | 0 | 0 |
| $E$ | [0,1150] | 1052.74 | 0 | 0 |
| Objective Function Value | | | | 575.6308 |
| Total Travel Distance | | | | 121.0070 |
| Total Delay | | | | 212.6099 |

(a) $\alpha = 3, \beta = 1$

| Node | Time Window | Start Time | Load | Delay |
|---|---|---|---|---|
| $S$ | [0,1150] | 0.00 | 0 | 0 |
| $P_5$ | [25,105] | 25.00 | 10 | 0 |
| $P_1$ | [31,226] | 123.06 | 20 | 0 |
| $P_2$ | [138,320] | 220.34 | 50 | 0 |
| $P_4$ | [317,550] | 330.44 | 80 | 0 |
| $D_2$ | [358,528] | 426.27 | 50 | 0 |
| $D_1$ | [313,545] | 542.22 | 40 | 0 |
| $D_4$ | [561,721] | 650.65 | 10 | 0 |
| $P_3$ | [337,595] | 744.26 | 20 | 149.26 |
| $D_3$ | [569,819] | 848.40 | 10 | 29.40 |
| $D_5$ | [800,960] | 951.21 | 0 | 0 |
| $E$ | [0,1150] | 1166.29 | 0 | 0 |
| Objective Function Value | | | | 687.7147 |
| Total Travel Distance | | | | 151.7285 |
| Total Delay | | | | 178.6620 |

(b) $\alpha = 1, \beta = 3$

routes are visualized in Figure 3.

From the result, parameter setting (a) gives an optimal solution with three locations violating the time window constraints and the total delay is higher compared with setting (b). However, the route produced by setting (b) has higher total travel distance. As the two objectives are incompatible with each other, the choice of parameters imposes the priority of the objective. When $\alpha$ is large, solutions with shorter distance are preferable. Under the condition where cost matters more than customer satisfaction, delay would be introduced if necessary to keep the distance short. During the process, priority for members is enforced as the solution would sacrifice the service quality of non-members by increasing their delay while keeping the time requirements for members satisfied. On the contrary, discrepancies between customers are reduced when $\beta$ value is large because time windows are nearly hard for non-members when greater penalty for delay is imposed in the objective.

Balance between travel cost and customer satisfaction can be achieved by adjusting the parameters. The phenomenon described above is highly desirable as service providers can apply this model tailored to their requirements and give an approximation for the pricing.

# 4 Mathematical Model

The problem is represented by a graph with nodes $V = \{0, 1, 2, ..., 2n + 1\}$. Each node represents a location which is either a pickup location, a customer delivery location or a depot. All vehicles start from depot $0$ and end at depot $2n + 1$. Let $P = \{1, 2, ...n\}$ represent the pickup nodes, $D = \{n + 1, n + 2, ...2n\}$ represent the delivery nodes and $N = P \cup D$ represent all nodes except the depots. Each request is given by a pair of nodes $(i, n + i)$. Let $q_i$ be the demand at node $i$ satisfying the followings: $q_i > 0$ for pickup nodes, $q_i < 0$ for delivery nodes, and $q_i = -q_{i+n}$ for $\forall i \in P$. Binary variable $m_i$ takes value 1 if and only if the request associated with node $i$ is ordered by a member, and is 0 otherwise. we introduce $V_m = \{i \in V \mid m_i = 1\}$ as the set of nodes associated with members' requests together with two depots.

At each node $i \in V$, the time window is denoted by $[a_i, b_i]$ and the service time by $s_i$. Let $d_{ij}$ be the distance between $i \in V$ and $j \in V$ and $t_{ij}$ be the travel time needed to travel from $i$ to $j$. We assume $d_{ij}$ and $t_{ij}$ both satisfy the triangular inequality.

Let $K$ be the set of vehicles and $Q^k$ be the maximum load capacity for vehicle $k \in K$.

Let $x_{ij}^k$ be a binary variable which is equal to 1 if and only if vehicle $k$ traversed directly from node $i$ to node $j$, and is 0 otherwise. Let non-negative $Q_i^k$ indicate the current load for vehicle $k$ after visiting node $i$. Let non-negative $T_i^k$ be the start service time for vehicle $k$ at node $i$. Let non-negative $DL_i^k$ indicate the delay of service at node $i$. $Q_i^k$, $T_i^k$ and $DL_i^k$ are well-defined only when the route of vehicle $k$ includes $i$.

A mathematical model can be proposed as

$$min\ \alpha \sum_{k \in K} \sum_{i,j \in V} d_{ij} x_{ij}^k + \beta \sum_{k \in K} \sum_{i \in V} DL_i^k \tag{1}$$

subject to

$$\sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1, \forall i \in P, \forall k \in K \tag{2}$$

$$\sum_{j \in V} x_{ji}^k - \sum_{j \in V} x_{ij}^k = 0, \forall i \in N, \forall k \in K \tag{3}$$

$$\sum_{j \in P \cup \{2n+1\}} x_{0j}^k = 1, \forall k \in K \tag{4}$$

$$\sum_{i \in D \cup \{0\}} x_{i,2n+1}^k = 1, \forall k \in K \tag{5}$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{j,n+i}^k = 0, \forall i \in P, \forall k \in K \tag{6}$$

$$Q_i^k + q_j - Q_j^k \leq M(1 - x_{ij}^k), \forall i, j \in V, \forall k \in K \tag{7}$$

$$q_i \leq Q_i^k \leq Q^k, \forall i \in P, \forall k \in K \tag{8}$$

$$0 \leq Q_i^k \leq Q^k + q_i, \forall i \in D, \forall k \in K \tag{9}$$

$$Q_0^k = 0, \forall k \in K \tag{10}$$

$$T_i^k + s_i + t_{ij} - T_j^k \leq M(1 - x_{ij}^k), \forall i, j \in V, \forall k \in K \tag{11}$$

$$T_i^k + t_{i,n+i} + s_i \leq T_{n+i}^k, \forall i \in P, \forall k \in K \tag{12}$$

$$T_i^k \geq a_i, \forall i \in V, \forall k \in K \tag{13}$$

$$T_i^k \leq b_i, \forall i \in V_m, \forall k \in K \tag{14}$$

$$T_i^k - b_i - DL_i^k \leq M(1 - \sum_{j \in N \cup \{0\}} x_{ji}^k), \forall i \in V \backslash V_m \tag{15}$$

$$x_{ij}^k \in \{0, 1\}, \forall i, j \in V, \forall k \in K \tag{16}$$

$$Q_i^k \geq 0, T_i^k \geq 0, DL_i^k \geq 0, \forall i \in V, \forall k \in K \tag{17}$$

The objective function minimizes the weighted sum of travel distance and delay penalty.

Constraints (2) ensure that each request is visited once. Constraints (3) ensure flow conservation and route continuity, which means that if vehicle enters node $i \in N$ it must also leave it. Constraints (4) and (5) ensure that each vehicle leaves start depot and arrives at end depot. Constraints (6) ensure that the pickup node and delivery node associated with the same request are visited by the same vehicle.

Constraints (7) guarantee that the load of each vehicle is updated correctly along its route where $M$ is a sufficiently large number, i.e. $max(0, Q_i^k + q_j - Q_j^k)$. Constraints (8) and (9) ensure that each vehicle would not violate the maximum capacity constraints at pickup and delivery nodes respectively. Constraints (10) require the initial load for each

vehicle to be $0$.

Constraints (11) ensure that the arrival time of vehicle at successor node is at least the sum of the arrival time and service time at predecessor node, along with the travel time in between, where $M$ is a sufficiently large number, i.e. $max(0, b_i + s_i + t_{ij} - a_j)$. Constraints (12) make sure that pickup node is serviced before delivery node associated with the same request. Constraints (13) ensure that the start service time at each node is after the start time window. Constraints (14) guarantee that no end time window is violated for requests by members. Constraints (15) ensure that the delay $DL_i^k$ at node $i$ is calculated correctly, where $M$ is a sufficiently large number, i.e. $max(0, b_{2n+1} - b_i)$.

# 5 Methodology

In this section, the algorithm proposed to solve our model is discussed. A modified version of the Adaptive Large Neighborhood Search (ALNS) based on Ropke and Pisinger (2006) was introduced and applied in this work.

## 5.1 ALNS Framework

The ALNS framework is based on the Large Neighborhood Search (LNS), which was proposed by Shaw (1998). The LNS heuristic starts with a feasible initial solution and gradually improves the objective value by applying one removal heuristic and one insert heuristic to the solution in each iteration. By removing a set of nodes and reinserting them optimally to the solution, the LNS searches a small fraction of solutions in a larger neighborhood space compared with traditional local search heuristics. The LNS was applied to the VRPPDTW and demonstrated good performance in Bent and Van Hentenryck (2006). However, the solution quality of the LNS is largely dependent on the quality of the insert heuristics. In previous works, branch-and-bound search was used as the insert heuristic. Ropke and Pisinger (2006) proposed the ALNS algorithm that allows the use of simple and fast insertion heuristics while achieving comparable performance to the LNS. The ALNS

framework chooses heuristics in each iteration from a set of removal heuristics and insert heuristics instead of using fixed ones as in the LNS. According to Pisinger and Ropke (2007), the ALNS was proved to be robust for different variants of the VRP, including the VRPPDTW.

We proposed the ALNS framework in this work based on Ropke and Pisinger (2006) with two modifications: (1) Shaw Removal heuristic was extended to take membership status into consideration; (2) acceptance criterion from Linear Threshold Acceptance was applied instead of criterion from Simulated Annealing.

The ALNS framework is described in Algorithm 1. In line 1, a feasible initial solution is constructed and the construction method is explained in Section 5.3.3. In line 2-3, a set of removal and insert operators are initialized. A total of $maxIter$ iterations are repeated in the algorithm. In each iteration, the ALNS removes and reinserts $q$ requests from the solution in line 7-8. The rules are defined by heuristics discussed in Section 5.2 and 5.3 respectively. In line 9-11, if new solution $s'$ has better objective than the best-so-far solution $s^*$, $s^*$ is updated accordingly. In line 12-14, if acceptance criterion is met, the current solution $s$ is updated. The criterion is discussed in Section 5.4. The whole searching process is divided into segments, each composed of $segIter$ iterations. In line 16, operator updates its score at the end of each segment. The rules for score adjustment and operator selection (line 6) are discussed in Section 5.5.

## 5.2 Removal Heuristics

This section describes 3 different removal heuristics. They each describes a different criterion to choose the requests to be removed and specifies the function $Remove(s, q)$ in Algorithm 1. The function takes in a feasible solution $s$ and an integer $q$, outputs the partial solution $s'$ with $q$ requests removed and the removed requests $R_{removed}$.

**Algorithm 1** ALNS Framework

$s^* \leftarrow$ feasible initial solution
$REM \leftarrow$ set of removal operators
$INS \leftarrow$ set of insert operators
$q \leftarrow$ number of requests to be removed
**for** ($i := 0; i < maxIter; i$++) **do**
  choose removal and insert operator from $REM$ and $INS$
  $s', R_{removed} \leftarrow Remove(s, q)$
  $s' \leftarrow Insert(s', R_{removed})$
  **if** $f(s') < f(s^*)$ **then**
    $s^* = s'$
  **end if**
  **if** acceptance criterion met **then**
    $s \leftarrow s'$
  **end if**
  **if** $maxIter$ mod $segIter == 0$ **then**
    update scores for operators in $REM$ and $INS$
  **end if**
**end for**
**return** $s^*$

### 5.2.1 Random Removal

Each time one random request is removed from the solution until the total number of removed requests reaches $q$.

### 5.2.2 Shaw Removal with Priority

Compared with removing one or two requests from each route in the solution, it is preferable to remove requests that are "close" to each other simultaneously. Shaw Removal defines the measure of similarity between two pairs of requests, and intends to remove requests that are more similar within one iteration. Based on Ropke and Pisinger (2006), a modified definition of similarity $R(i, j)$ between request $i$ and $j$ is presented below

$$R(i,j) = \omega_1 \frac{d_{p_i,p_j} + d_{d_i,d_j}}{d_{max}} + \omega_2 \frac{|T_{p_i} - T_{p_j}| + |T_{d_i} - T_{d_j}|}{T_{max}} + \omega_3 \frac{|q_i - q_j|}{q_{max}} + \omega_4 |m_i - m_j| \quad (18)$$

where $p_i$ and $d_i$ are the pickup location and delivery location associated with request $i$ respectively, and variables $d_{ij}, T_i, q_i$ have the meaning specified in Section 4.

$R(i, j)$ is the weighted sum of four terms, where each term measures the similarity from a different aspect. The first term describes the spatial similarity between two requests represented by the total distance between two pickup locations and two delivery locations. The second term describes the temporal similarity measured by the total difference of visiting times. The third term describes the similarity with respect to difference in load. The fourth term considering membership status of requests is added for our model. Its value is 0 if both requests are ordered by members or both by non-members, so a pair of requests would be defined as more similar if they share the same membership status. Note that by setting $\omega_4 = 0$, membership status is not taken into consideration.

The contributions of each aspect to $R(i, j)$ are controlled by coefficients $\omega_1, \omega_2, \omega_3, \omega_4$. To deal with the difference in scale between the first three terms, the maximum distance between all pairs of locations $d_{max}$, the maximum time window size $T_{max}$ and the maximum load among all requests $q_{max}$ are used to normalize the terms to take values on $[0, 1]$. Two requests $i, j$ are considered more related if their similarity $R(i, j)$ is smaller.

The heuristic is explained in Algorithm 2. In line 2, a random request is chosen to initialize $R_{removed}$, which is the set of requests to be removed from the current solution. In each iteration, one random request $r'$ is first selected from $R_{removed}$ in line 4. The similarity measurement between every unselected request $r$ and $r'$ is then calculated and the results are sorted by ascending order in line 5-6. In line 8-9, the request with $(y^p|L|)^{th}$ smallest similarity measurement is added to $R_{removed}$. Parameter $p \geq 1$ controls the randomness in the removal heuristic. Choosing smaller $p$ increases randomness because requests that are not so related may be selected as $y^p$ gets larger with fixed $y$. Note that Random Removal is one special case of Shaw Removal when $p = 1$. In line 11, solution $s'$ is generated by removing requests in $R_{removed}$ from $s$ and updating information for all locations.

**Algorithm 2** Shaw Removal with Priority (Solution $s$, $q$)

1: $r \leftarrow$ request randomly selected from $s$
2: $R_{removed} \leftarrow \{r\}$
3: **while** $size(R_{removed}) < q$ **do**
4: $r' \leftarrow$ request randomly selected from $R_{removed}$
5: $L \leftarrow$ list of request $r$ not in $R_{removed}$
6: $L \leftarrow L$ sorted by ascending $R(r, r')$
7: $y \leftarrow Random(0, 1)$
8: $r \leftarrow$ request at $y^p|L|$ position in $L$
9: $R_{removed} \leftarrow R_{removed} \cup \{r\}$
10: **end while**
11: $s' \leftarrow Remove(s, R_{removed})$
12: **return** $s', R_{removed}$

### 5.2.3 Worst Removal

The cost of request $r$ is defined as the difference between the total objective with and without the pickup-delivery pair associated with $r$. It is denoted as $\Delta f_r = f(s) - f_{-r}(s)$, where $s$ is the current solution, $f$ is the objective function, and $f_{-r}(s)$ is the objective after removing request $r$ from $s$. Larger $\Delta f_r$ value indicates that request $r$ is currently placed at a bad position, and it is highly possible to find an improved solution after replacing such requests. Therefore, Worst Removal is designed to find and remove requests with large $\Delta f_r$ in each iteration. The details are explained in Algorithm 3. The algorithm is similar to Shaw Removal with Priority except that the requests are sorted by descending order of request cost in each iteration (line 3-4).

Similar to Shaw Removal with Priority, parameter $p$ is included to introduce randomness in the algorithm. Without the presence of randomness, Worst Removal would keep trying removing the same set of requests over consecutive iterations. The performance of the heuristic would decrease because it could not effectively explore the neighborhood space.

**Algorithm 3** Worst Removal (Solution $s$, $q$)

1: $R_{removed} \leftarrow \emptyset$
2: **while** $size(R_{removed}) < q$ **do**
3: $L \leftarrow$ list of request $r$ not in $R_{removed}$
4: $L \leftarrow L$ sorted by descending $\Delta f_r$
5: $y \leftarrow Random(0, 1)$
6: $r \leftarrow$ request at $y^p|L|$ position in $L$
7: $R_{removed} \leftarrow R_{removed} \cup \{r\}$
8: **end while**
9: $s' \leftarrow Remove(s, R_{removed})$
10: **return** $s', R_{removed}$

## 5.3 Insert Heuristics

This section describes different insert heuristics. Each of them introduces one rule to determine positions for requests to be inserted and specifies the function $Insert(s', R_{removed})$ in Algorithm 1. The heuristics take in a partial solution and a set of requests, and output a solution with requests reinserted.

### 5.3.1 Simple Greedy Insert

The idea of Simple Greedy Insert is to insert the "best" request in the sense of objective increase in each iteration. Let $\Delta f_{rk}$ denote the insertion cost as the minimum objective increase by inserting request $r$ on route $k$. The algorithm is shown in Algorithm 4. In line 2-6, insertion cost is calculated for each request and each route. In line 10-12, request $r^*$ with minimum objective increase is inserted in the partial solution and removed from the set of unprocessed requests. In line 14-18, if not all requests are inserted, empty solution is returned to indicate that the heuristic fails to reconstruct a feasible solution, otherwise $s'$ is the desired solution.

**Algorithm 4** Simple Greedy Insert (Solution $s'$, $R_{removed}$)

**while** $R_{removed} \neq \emptyset$ **do**
  **for** each request $r$ in $R_{removed}$ **do**
    **for** each route $k$ in solution $s$ **do**
      calculate $\Delta f_{rk}$
    **end for**
  **end for**
  **if** no feasible insertion **then**
    **break**
  **end if**
  $r^* \leftarrow \underset{r \in R_{removed}}{argmin} (\underset{k \in K}{min} \Delta f_{rk})$
  insert request $r^*$ in route $k$ at position with minimum objective value increase
  remove request $r^*$ from $R_{removed}$
**end while**
**if** $R_{removed} = \emptyset$ **then**
  **return** $s'$
**else**
  **return** $\emptyset$
**end if**

### 5.3.2 Regret Insert

The Simple Greedy Insert heuristic is myopic as it only considers one step after insertion. It has the problem of leaving "bad" requests and having them inserted with comparatively high cost at the end of the process when there is no choice left. The Regret Insert aims to solve this problem by taking more steps into consideration. Specifically, the Regret-$m$ Insert considers $m$ further steps.

For fixed route $k$ and request $r$, let best insertion position denote the position at which inserting $r$ introduces minimum objective increase among all positions in $k$. Denote the associated objective increase as best insertion cost. Let $k_{rn}$ denote the route with $n^{th}$ lowest best insertion cost among all routes, and let $\Delta f_{r,k_{rn}}$ denote the associated best insertion cost. If request $r$ has no feasible insertion position on route $k_{rn}$, we define $\Delta f_{r,k_{rn}} = M$, where $M$ is a sufficiently large constant. In the Regret-$m$ Insert heuristic, the regret value

for request $r$ is defined as

$$c_r = \sum_{n=1}^{m} (\Delta f_{r,k_{rn}} - \Delta f_{r,k_{r1}}) \tag{19}$$

The regret value describes the level of difficulty of leaving the request uninserted and having it inserted in further iterations. The heuristic attempts to insert request that maximizes the regret value. Ties are broken by prioritizing requests with lower insertion cost. Note that the Simple Greedy Insert is the Regret-1 Insert because of the tie-breaking rule. By setting $\Delta f_{r,k_{rn}} = M$, for requests that have feasible insertion routes on less than $m$ routes, requests with fewer feasible insertion routes are selected.

The general procedure is similar to Simple Greedy Insert in Algorithm 4 where in line 10 we select the request that maximizes the regret value $c_r$ instead.

In this work, Regret-1 (Simple Greedy), Regret-2, Regret-3, Regret-4, Regret-$|K|$ were used as the set of insertion operators.

### 5.3.3 Initial Insert

The initial solution construction is completed by running the Simple Greedy Insert heuristic with an empty solution and all requests as input.

## 5.4 Acceptance Criterion from Linear Threshold Acceptance (TA)

The acceptance criterion used in this work is from linear TA with end temperature set to 0. $T$ is the temperature serving as the threshold for acceptable objective gap. In each iteration, improved neighboring solution is always accepted and deteriorated solution would be accepted if $\frac{f(s')-f(s^*)}{f(s^*)} < T$ is satisfied. $T$ is initialized as $T_{start}$ and is decreased by cooling rate $\Delta T = \frac{T_{start}}{maxIter}$ in each iteration.

Linear TA was first proposed by Dueck and Scheuer (1990) and was proved to be more computationally effective and be able to achieve solutions with good quality. Another popular choice for acceptance criterion is from simulated annealing (SA) where better so-

lutions are always accepted and deteriorated solutions would be accepted with a probability of $e^{-\frac{f(s')-f(s)}{T}}$. According to Santini et al. (2018), the performance of both criteria are comparable when applied in the ALNS framework. Acceptance criterion from TA was selected for this work because (1) the acceptance is determined by comparison with threshold $T$ without the trouble to generate random numbers and compute probability; (2) only one parameter $T_{start}$ needs to be tuned as the cooling rate can be calculated directly.

## 5.5 Adaptive Weight Adjustment

The same mechanism was applied in this work as in Ropke and Pisinger (2006). Readers may refer to their work for more details.

For each operator $i$ in removal operator set and insert operator set, weight $w_i$ is assigned. Operator selection is performed with roulette wheel selection principle such that operator $i$ has the probability $\frac{w_i}{\sum_{i \in O} w_i}$ of being selected where $O = REM$ or $O = INS$.

The score $\Pi_i$ for operator $i$ is updated in each iteration by the amount determined by parameters $\sigma_1, \sigma_2, \sigma_3$ at different scenarios:

1. If the new solution introduces a new global best-so-far solution, $\sigma_1$ is added;
2. If the new solution improves objective value and was not accepted before, $\sigma_2$ is added;
3. If the new solution was not accepted before, and is accepted with a deteriorated objective, $\sigma_3$ is added.

The whole searching process is divided into segments. At the start of each segment, scores for all operators are reset to 0. At the end of each segment, weights are updated using the formula

$$w_{i,j+1} = (1 - r)w_{ij} + r\frac{\Pi_i}{\theta_i} \tag{20}$$

where $w_{ij}$ is the weight for operator $i$ in segment $j$, $\Pi_i$ is the score operator $i$ obtained in this segment, and $\theta_i$ is the number of times that operator $i$ is used in this segment.

# 6 Computational Results

In this section, the ALNS configuration for our model and relevant results are reported. We also discuss the characteristics and performance of our ALNS framework.

Our algorithm was implemented in Java. Experiments were conducted on a personal computer running Windows 10 Pro with Intel(R) Core(TM) i7-8565U CPU and 16.0 GB RAM.

## 6.1 Data Generation

In the experiments, instances used were modified based on the data set proposed by Li and Lim (2003). The data set describes single-depot pickup and delivery problems with capacity constraints, precedence constraints and time window constraints. It contains six classes: LC1, LC2, LR1, LR2, LRC1, LRC2. The customers are clustered in LC instances, and are randomly distributed in LR instances. In LRC instances, the distributions are a combination of both. For LC1, LR1 and LRC1 instances, the optimal solutions contain a larger number of vehicles. On the contrary, LC2, LR2 and LRC2 instances can be solved with a significantly lower number vehicles and are more computationally expensive. The data set is available at SINTEF (2008).

The data set used in this work was generated based on 33 Li and Lim instances, among which 12 instances are from LC1 (size 100), 11 are from LC2 (size 100) and 10 are from LC1 (size 200). To investigate the behavior of the solution with different sets of weights in the objective function, only instances in class LR with randomly distributed customers were selected.

Our problem differs from the original data set in that time window violation is allowed. To better investigate the trade-off between distance and delay, time windows were shrinked as illustrated below.

Let $u$ denote the shrink percentage of the time window size. Given $u$ and time window $[a, b]$, the shrinked time window is given by $[a + u(b - a), b - u(b - a)]$. For the depots, the

Table 3: Maximum Time Window Shrink Percentage $u$

| Test Instance | $u$ |
|---|---|
| LR101 | 0.034 |
| LR102 | 0.023 |
| LR103 | 0.004 |
| LR104 | 0.000 |
| LR105 | 0.016 |
| LR106 | 0.094 |
| LR107 | 0.096 |
| LR108 | 0.092 |
| LR109 | 0.015 |
| LR110 | 0.184 |
| LR111 | 0.077 |
| LR112 | 0.401 |

(a) size 100 LR1

| Test Instance | $u$ |
|---|---|
| LR201 | 0.057 |
| LR202 | 0.100 |
| LR203 | 0.119 |
| LR204 | 0.269 |
| LR205 | 0.091 |
| LR206 | 0.197 |
| LR207 | 0.207 |
| LR208 | 0.280 |
| LR209 | 0.157 |
| LR210 | 0.074 |
| LR211 | 0.419 |

(b) size 100 LR2

| Test Instance | $u$ |
|---|---|
| LR1_2_1 | 0.006 |
| LR1_2_2 | 0.125 |
| LR1_2_3 | 0.070 |
| LR1_2_4 | 0.105 |
| LR1_2_5 | 0.052 |
| LR1_2_6 | 0.065 |
| LR1_2_7 | 0.082 |
| LR1_2_8 | 0.031 |
| LR1_2_9 | 0.069 |
| LR1_2_10 | 0.005 |

(c) size 200 LR1

end time window is decreased by $0.1u$ percent and the start time window is not modified. $u$ is initialized to $0$. In each iteration, $u$ is increased by $0.001$. The process continues until $u$ cannot be further increased to produce a feasible solution for instances with time window shrinked. The maximum shrink percentages found are reported in Table 3.

In addition, we adapted the data set by assigning membership status for each pair of pickup and delivery locations in correspondence to our model. Let $m_p$ represent the percentage of requests ordered by members. Each request was randomly chosen to have membership status with probability $m_p$. The pair of pickup-delivery locations were set to have the same membership status as the associated request.

The data set in this work is composed of 99 instances. 3 instances were generated by adding membership status with $m_p = 0.3, 0.5, 0.7$ for each of the 33 selected instances with shrinked time windows.

## 6.2 Parameter Tuning and Selection

In this section, parameters used in the experiments are presented and tuning process is explained.

### 6.2.1 Parameters

In the ALNS framework, $maxIter = 25000$, $segIter = 100$, $r = 0.1$, $\sigma_1 = 33$, $\sigma_2 = 9$, $\sigma_3 = 13$ were applied according to Ropke and Pisinger (2006). Initial temperature $T_{start}$ was tuned as explained in Section 6.2.2, and $T_{start} = 0.02$ was used in the experiments.

For removal heuristics, $q$ was chosen to be a random number between $[0.2n, 0.4n]$ where $n$ is the total number of requests. Each time before the removal operator starts to remove requests, a different random $q$ is generated. For both Shaw Removal with Priority and Worst Removal, $p$ was set to 6. For Shaw Removal with Priority, $\omega_1 = 9$, $\omega_2 = 3$, $\omega_3 = 2$ were set based on Ropke and Pisinger (2006). Parameter $\omega_4$ was tuned as explained in Section 6.2.2, and $\omega_4 = 1.0$ was applied.

### 6.2.2 Parameter Tuning

During tuning process, 6 instances were used, 2 from each instance class with $m_p = 0.5$. Selected tuning instances are LR110, LR111, LR202, LR203, LR1_2_2, LR1_2_7.

The general tuning process is explained as follows. For each parameter, tuning is performed with other parameters fixed. A list of possible parameter values to be compared is chosen. For each value in the list, experiments are repeated $n$ times with model configured with specified parameter value. We define $gap = \frac{f - f_{min}}{f_{min}}$ to measure the objective deviation, where $f$ is the objective value for this run and $f_{min}$ is the minimum objective value for the instance found in the tuning process. For each instance, the average gap value is calculated as the average of $gap$ among $n$ runs. The total amount of objective deviation $gap_{total}$ for each parameter value is calculated by summing up the average gap for all tuning instances. The overall performance of each parameter value is measured by the total

deviation $gap_{total}$. Value associated with the smallest deviation is selected and fixed for the parameter, and the tuning process moves on to the next parameter. The process stops after a single pass when all parameters are tuned once.

In the tuning process, $n$ was set to 5. Parameters except $T_{start}$ and $\omega_4$ were set as previously stated. Objective weights in our model were set to $\alpha = 1, \beta = 1$.

Table 4: $T_{start}$ Tuning

| | LR110 | LR111 | LR202 | LR203 | LR1_2_2 | LR1_2_7 |
|---|---|---|---|---|---|---|
| $f_{min}$ | 1402.427458 | 1304.919804 | 1388.315765 | 1140.543174 | 4261.756804 | 3213.519512 |

(a) minimum objective value

| $T_{start}$ | LR110 | LR111 | LR202 | LR203 | LR1_2_2 | LR1_2_7 | $gap_{total}$ |
|---|---|---|---|---|---|---|---|
| 0.01 | 0.000000 | 0.002723 | 0.007530 | 0.019600 | 0.000052 | 0.005380 | 0.035285 |
| 0.02 | 0.000266 | 0.000670 | 0.004127 | 0.003346 | 0.001054 | 0.005944 | **0.015407** |
| 0.04 | 0.000518 | 0.000000 | 0.002781 | 0.007997 | 0.001612 | 0.008621 | 0.021530 |
| 0.06 | 0.001325 | 0.000205 | 0.003197 | 0.005056 | 0.001327 | 0.009969 | 0.021078 |
| 0.08 | 0.001443 | 0.002010 | 0.003768 | 0.010860 | 0.002395 | 0.010998 | 0.031474 |

(b) average gap

Tuning for $T_{start}$ was conducted first with $\omega_4$ set to 0. $T_{start}$ was tested for values in $[0.01, 0.02, 0.04, 0.06, 0.08]$. The result is shown in Table 4. From the $gap_{total}$ column, $T_{start} = 0.02$ with the smallest deviation was selected for further tuning use.

$\omega_4$ was tuned next with $T_{start}$ set to $0.02$. The sequence of parameter values for $\omega_4$ was chosen as $[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0]$. Similarly, average gaps were calculated and the results are shown in Table 5. Parameter $\omega_4 = 1.0$ with minimum $gap_{total}$ was selected.

Note that when $\omega_4 = 0$, Shaw Removal with Priority does not consider difference in membership status in the similarity definition. From the $gap_{total}$ column we can see that Shaw Removal with Priority has slightly better performance with the additional term when solving our model.

Table 5: $\omega_4$ Tuning

| | LR110 | LR111 | LR202 | LR203 | LR1_2_2 | LR1_2_7 |
|---|---|---|---|---|---|---|
| $f_{min}$ | 1402.427458 | 1304.919804 | 1387.375228 | 1140.543174 | 4261.756804 | 3213.997285 |

(a) minimum objective value

| $T_{start}$ | LR110 | LR111 | LR202 | LR203 | LR1_2_2 | LR1_2_7 | $gap_{total}$ |
|---|---|---|---|---|---|---|---|
| 0.0 | 0.001734 | 0.000205 | 0.006600 | 0.020644 | 0.000286 | 0.005854 | 0.035322 |
| 0.5 | 0.000813 | 0.000000 | 0.003838 | 0.011476 | 0.000000 | 0.007236 | 0.023363 |
| 1.0 | 0.000174 | 0.000000 | 0.002108 | 0.001195 | 0.000156 | 0.004011 | **0.007644** |
| 1.5 | 0.000000 | 0.000205 | 0.003834 | 0.007425 | 0.000196 | 0.005313 | 0.016973 |
| 2.0 | 0.000113 | 0.000000 | 0.002824 | 0.010950 | 0.000976 | 0.005401 | 0.020264 |
| 2.5 | 0.001046 | 0.000742 | 0.002075 | 0.026600 | 0.002067 | 0.003422 | 0.035952 |
| 3.0 | 0.000113 | 0.000670 | 0.002793 | 0.012454 | 0.000052 | 0.004379 | 0.020461 |
| 3.5 | 0.000000 | 0.000670 | 0.003592 | 0.009695 | 0.000000 | 0.006460 | 0.020417 |
| 4.0 | 0.000560 | 0.000031 | 0.005469 | 0.010157 | 0.000000 | 0.008297 | 0.024514 |
| 5.0 | 0.000427 | 0.000670 | 0.003692 | 0.010289 | 0.000248 | 0.008250 | 0.023575 |
| 6.0 | 0.000466 | 0.000000 | 0.007074 | 0.018956 | 0.001054 | 0.005379 | 0.032928 |
| 7.0 | 0.000113 | 0.002010 | 0.001915 | 0.010646 | 0.000000 | 0.004372 | 0.019055 |

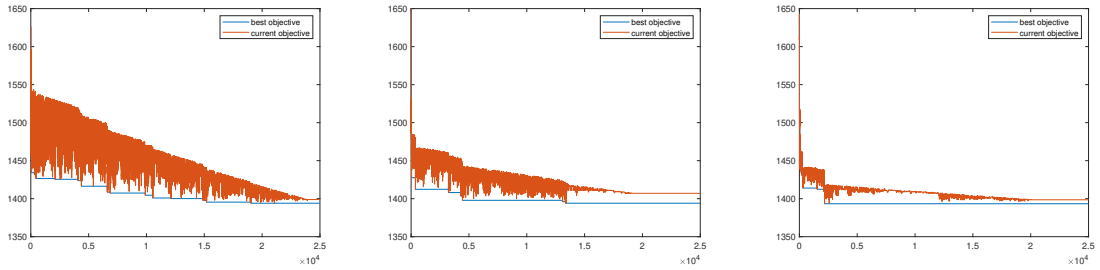
(b) average gap

## 6.3 ALNS Framework

In this section, the characteristics of our ALNS are analyzed and the robustness is tested.

### 6.3.1 Acceptance Criterion

The influence of the initial temperature on the searching process is investigated. In Figure 4, the objective of accepted solution (current objective) and the objective of best-so-far solution (best objective) are plotted. Several observations can be made. Firstly, the value of accepted objective value decreases linearly, which is resulted from the linear decrease of threshold $T$. Secondly, global best objective value is updated each time when two lines coincide. The update happens more frequently at the beginning of the search process when the solution quality is not good. Thirdly, the local search process accepts fewer solutions when $T_{start}$ is lower. It is shown by the increasing sparsity from $(a)$ to $(c)$. Lower threshold value would require solutions to have a smaller objective gap to be accepted.




(a) $T_{start} = 0.08$ (b) $T_{start} = 0.04$ (c) $T_{start} = 0.02$

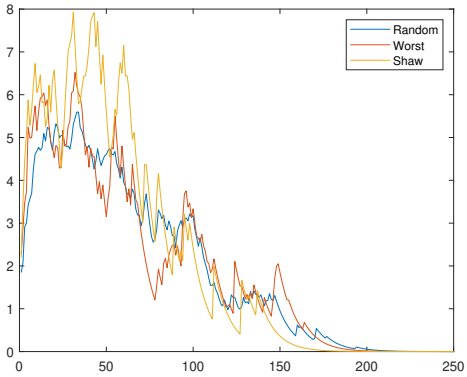
Figure 4: Objective change of best-so-far solution and current accepted solution over iterations at different initial temperature $T_{start}$ for instance LR202 (size 100)

### 6.3.2 Adaptive Weight Adjustment

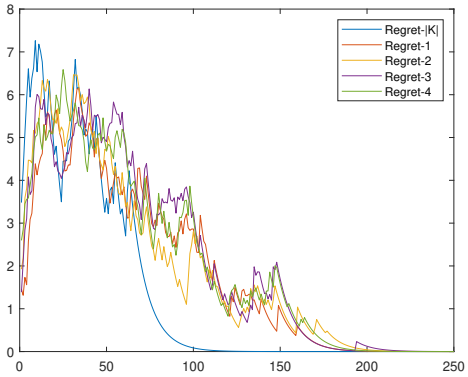
Instead of using a fixed set of operator weights, weights are updated based on previous operator performance in the ALNS framework. Higher weight is an indicator for better performance of the operator. As shown in Figure 5, different instances have different operator weight distributions. For removal heuristics, Shaw Removal with Priority has the best performance for LR110 while Random Removal works the best for LR1_2_2. For insert heuristics, Regret-$|K|$ Insert has the greatest performance for LR1_2_2 while has the worst performance for LR110. Therefore, the ALNS is able to remain robust over instances of different types because of its automatic weight adjustment ability.

### 6.3.3 Results for Li and Lim Benchmark Instances

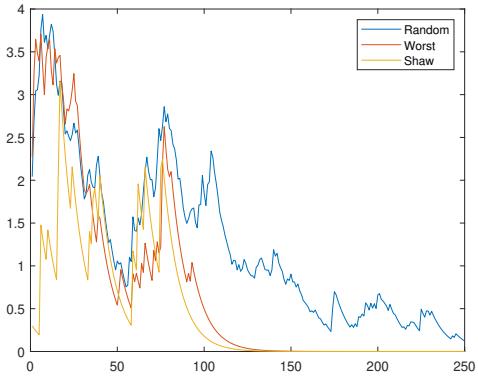
To test the robustness of our ALNS framework, experiments were performed on all instances of size 100 in the data set proposed by Li and Lim (2003). All requests were set to be ordered by members in our model, and $\omega_4$ was set to 0 in Shaw Removal with Priority. Results for one run are shown in Appendix A. Objective gap is calculated by $\frac{f - f_{optimal}}{f_{optimal}}$, where $f$ is the objective produced by our ALNS and $f_{optimal}$ is the optimal objective reported in the literature. Among all the 56 instances, only 9 produced objectives different from the optimal. 6 of them reported lower objectives with more vehicles used. It is reasonable as the minimization of vehicle number was not included as part of our objective.



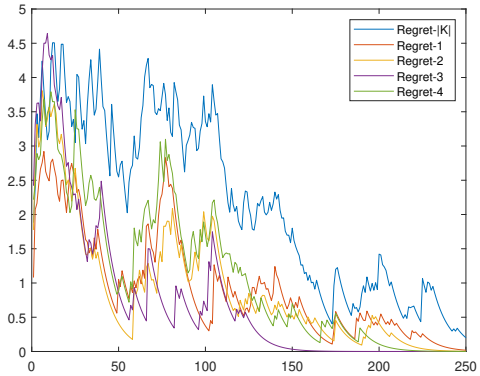

(a) Removal operator weights (LR110 size 100)




(b) Insert operator weights (LR110 size 100)




(c) Removal operator weights (LR1_2_2 size 200)




(d) Insert operator weights (LR1_2_2 size 200)

Figure 5: Operator weight change over iterations

Only 3 instances found solutions with deteriorated objective, but their objective gaps were all within 5%. Therefore, we conclude that our ALNS is robust for the VRPPDTW.

### 6.3.4 Comparison with CPLEX OPL solver

To investigate the performance of our ALNS framework for the VRPPDTW with Priority, instances with smaller size were generated and tested. Results were compared with those obtained from CPLEX OPL with a 60 minutes time limit. The small data set was generated by randomly extracting $\kappa$ pair of requests from the tuning instances. The instances were renamed by adding suffix "_$\kappa$". Parameters were configured as in Section 6.2

and objective weights in our model were set to $\alpha = 1, \beta = 1$. Experiments were conducted for instances with $\kappa = 5, 10, 15$. The results are shown in Table 6.

Table 6: Comparison between the ALNS and CPLEX OPL solver

CPLEX OPL Time marked with * reaches 60m time limit, CPLEX OPL Objective "_" represents no feasible solution found.

| Test Instance | Objective | | Time (s) | |
|---|---|---|---|---|
| | ALNS | CPLEX OPL | ALNS | CPLEX OPL |
| LR110_5 | 196.498707 | 196.498707 | 1.465 | 0.693 |
| LR111_5 | 265.646788 | 265.646788 | 1.229 | 0.480 |
| LR202_5 | 211.093519 | 211.093519 | 0.712 | 0.204 |
| LR203_5 | 275.334532 | 275.334532 | 1.001 | 0.305 |
| LR1_2_2_5 | 518.822142 | 518.822142 | 0.825 | 0.207 |
| LR1_2_7_5 | 356.406622 | 356.406622 | 2.229 | 0.506 |

(a) $\kappa = 5$

| Test Instance | Objective | | Time (s) | |
|---|---|---|---|---|
| | ALNS | CPLEX OPL | ALNS | CPLEX OPL |
| LR110_10 | 354.713923 | 354.713923 | 3.692 | 495.719 |
| LR111_10 | 375.992850 | 375.992850 | 3.093 | 1408.489 |
| LR202_10 | 433.965041 | 433.965041 | 4.499 | 307.115 |
| LR203_10 | 467.770907 | 473.629718 | 4.076 | *3600.000 |
| LR1_2_2_10 | 846.584194 | 846.584194 | 2.878 | 123.186 |
| LR1_2_7_10 | 630.045079 | 630.045079 | 4.190 | 39.552 |

(b) $\kappa = 10$

| Test Instance | Objective | | Time (s) | |
|---|---|---|---|---|
| | ALNS | CPLEX OPL | ALNS | CPLEX OPL |
| LR110_15 | 557.224122 | _ | 4.736000 | *3600.000 |
| LR111_15 | 554.769693 | 554.769693 | 4.834000 | 0.480 |
| LR202_15 | 611.232671 | _ | 13.553000 | *3600.000 |
| LR203_15 | 558.810047 | _ | 9.745000 | *3600.000 |
| LR1_2_2_15 | 949.364623 | 949.364623 | 6.100000 | 746.769 |
| LR1_2_7_15 | 1049.695084 | _ | 7.463000 | *3600.000 |

(c) $\kappa = 15$

When $\kappa = 5$, OPL outperformed the ALNS with respect to computational time, but the ALNS could find optimal solutions for all instances at a comparably fast speed. When $\kappa = 10$, the ALNS was able to find a better solution for LR203 within 5 seconds compared

with the best solution OPL found upon time limit reached. For other instances, both the ALNS and OPL were able to find the optimal solutions, but the ALNS was 9.43 to 455.38 times faster. When $\kappa = 15$, OPL was only able to solve two instances while the ALNS could find feasible solutions for all instances without much increase in computational time. Since for all the instances that OPL succeeded in finding the optimal objective, ALNS was able to produce the same result, we conclude that our ALNS is robust when applied to the model proposed in this work. In addition, the ALNS takes a considerably shorter amount of time than OPL during the experiments for $\kappa = 10, 15$, and the difference in computational time increases when the problem size increases. It is reasonable to assume that the ALNS is much more computationally effective in the following experiments of size 100 and 200.

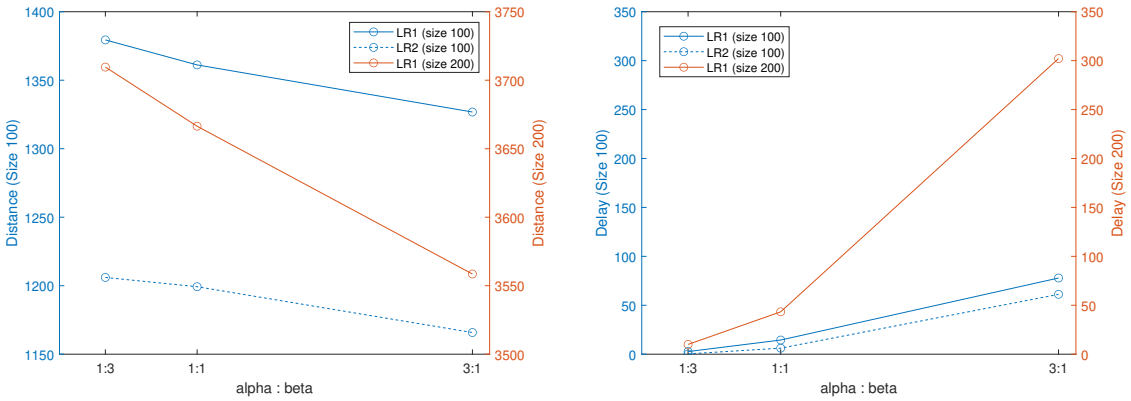
## 6.4 Results and Discussions

In this part, 3 sets of objective weights $\alpha = 1, \beta = 3; \alpha = 1, \beta = 1; \alpha = 3, \beta = 1$ were tested on our data set. For each instance, experiments were repeated 5 times for each set of objective weights. Average objectives were calculated over instances for each combination of instance class and member percentage $m_p$. The complete results are shown in Appendix B. The results are discussed as follows.

### 6.4.1 Comparison between different Objective Weights

For each set of objective weights, the aggregated objective for each instance class was calculated by averaging over instances in this class among all member percentages. Aggregated distance objective and delay objective change over the objective weights fraction $\frac{\alpha}{\beta}$ are plotted respectively in Figure 6. In general, distance decreases while delay increases as $\frac{\alpha}{\beta}$ increases. The behavior is resulted from the intention to minimize the part of the objective with higher weight. Two instance classes with size 100 exhibit similar trends while the one with size 200 exhibits more obvious objective difference for both distance and delay. We conclude that the characteristics of our model are more obvious as the problem size increases, while it is not influenced by vehicle number used in the solution. In addi-

tion, extreme behavior could be achieved by adjusting objective weights. Note that when $\alpha = 1, \beta = 3$, all instances display nearly no delay on average, resembling the behavior of models with only hard time windows. This behavior is useful in real life as stakeholders could choose to maximize customer satisfaction to the largest extent with the expense of distance cost regardless of the distribution of customer location and customer type.




(a) Aggregated distance

(b) Aggregated delay

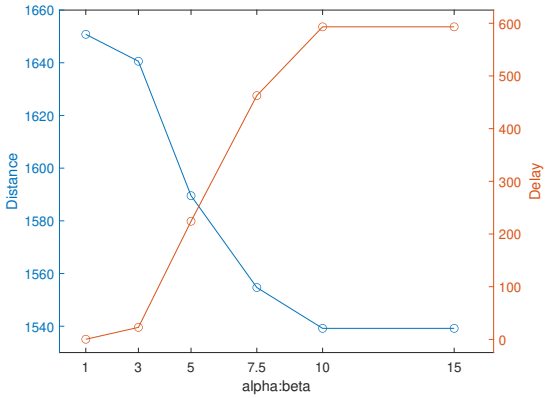
Figure 6: Aggregated objective change over objective weights fraction for LR1 (size 100), LR2 (size 100) and LR1 (size 200)

To further investigate the influence of objective weights, 3 instances LR101, LR209 and LR1_2_2 with $m_p = 0.5$ were selected to conduct experiments with more sets of objective weights. The result is shown in Figure 7. Several observations could be made from the figures:

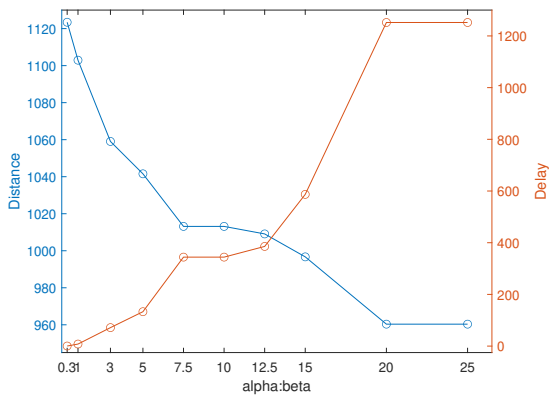
1. The distribution of the objective among distance and delay would remain stable when $\frac{\alpha}{\beta}$ is very large or very small. It could be explained as there would be lower bounds for both the distance and the delay for each instance to have feasible solutions;

2. All 3 instances have at least one "turning segment" where the objective distribution changes rapidly with only a small amount of increase in $\frac{\alpha}{\beta}$. The presence of turning segments shows that our model is very sensitive to the objective weights within a certain range;

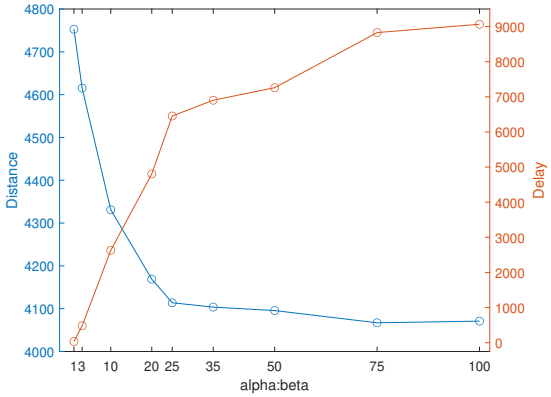
3. Objective weight ranges where turning segments occur and where the stable behaviors occur differ from instance to instance. The speed of objective change in turning segments




(a) Objective change (LR101)




(b) Objective change (LR209)




(c) Objective change (LR1_2_2)

Figure 7: Aggregated objective change over objective weights fraction for LR101, LR209 and LR1_2_2

is also dependent on the instance;

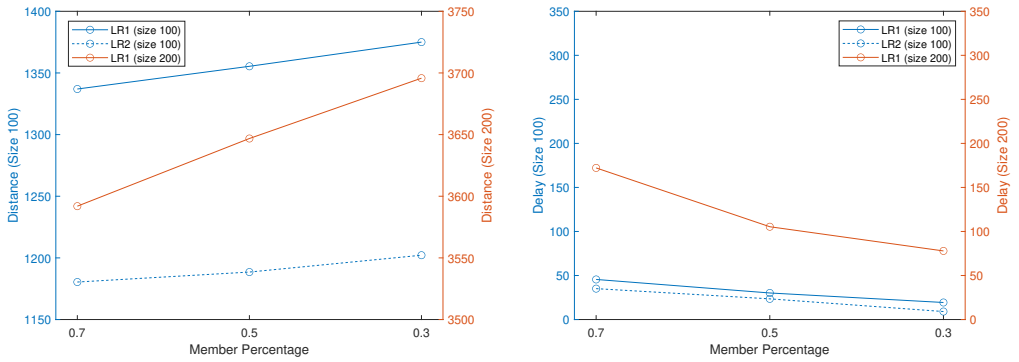
4.The delay increments are larger than the distance decrements for the same amount of increase of $\frac{\alpha}{\beta}$. Comparing the absolute value of the slope for the blue line and the red line, the speed of delay increase is around 11 times the speed of distance decrease as delay increases around 8000 while distance decreases only around 700 for LR1_2_2. Similarly, the factor is around 5 for both LR101 and LR209. Again, we conclude that the distribution of objective change in our model is more apparent for instances of larger size. We also conclude that the delay objective in our model is more sensitive to objective weights compared with the

distance objective.

The observations indicate that although distance would not fluctuate too much, small perturbations would yield solutions with quite different distributions of distance and delay in the objective. Therefore, the weights need to be tested and selected carefully for each case when our model is applied.

### 6.4.2 Comparison between different Member Percentages

In this subsection, we focus on results produced by data set with different member percentages $m_p$. For each $m_p$, the aggregated objective for each class was calculated by averaging over results from all experiments conducted on instances with member percentage $m_p$ in that class. The result is presented in Figure 8.

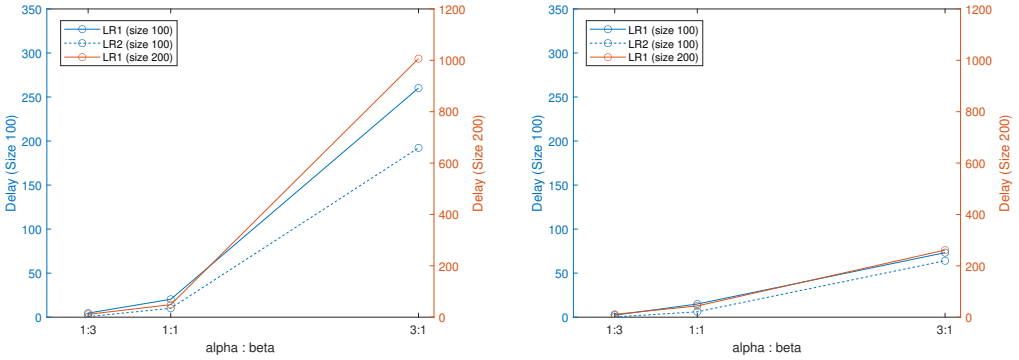



(a) Distance objective change (b) Delay objective change

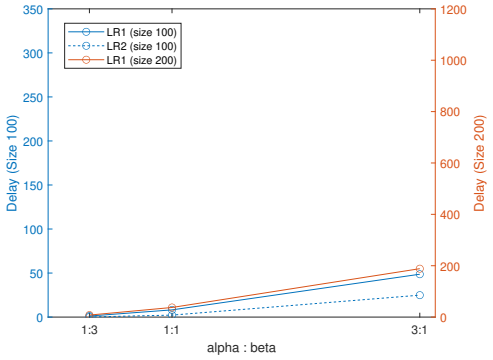
Figure 8: Objective change over different $m_p$

In general, instances with lower $m_p$ would have larger proportion of delay in the objective. For instances with the same data except $m_p$, those with higher $m_p$ have smaller feasible solution space as more time window constraints are presented in the model. Intuitively, optimal solutions with possible higher delay for instances with $m_p = 0.3$ may not be feasible for those with $m_p = 0.7$. Comparing the slope of the red line and the blue line, the behavior is more prominent for problem of larger size.

Furthermore, delay objective change over objective weights fraction is plotted for each




(a) $m_p = 0.3$ (b) $m_p = 0.5$




(c) $m_p = 0.7$

Figure 9: Delay objective change for different $m_p$

$m_p$ in Figure 9. One important observation is that instances with smaller $m_p$ would amplify the influence of objective weights change and encourage more extreme behavior. It could be explained as there would be less constraints and more flexibility for instances with smaller $m_p$. Instances with more non-members requests would require lower level of customer satisfaction in general.

# 7 Limitations and Future Work

Several improvements could be made for our model:

1. As shown in the experiment results, our model would generate solutions with extremely

high delay when the weight imposed on distance objective is too large. In addition, delay objective increases much faster than distance objective. Since we proposed the model to find balanced solutions between cost and customer satisfaction for stakeholders, increasing the delay of non-members to the unacceptable amount to decrease a small amount of distance cost may not be the desired behavior. The problem is resulted from the constant weights used in our model as it fixed the penalty regardless of the delay for the customers. It is reasonable to propose a model using a function that increases when delay increases as the weight for delay objective. Different functions could be tested and compared;
2. The cost of vehicles could be incorporated as part of the objective;
3. In this work, only members and non-members were considered. However, it is easy to extend our model with customers of more than two categories by assigning different weights to delay penalties of different categories.

For our proposed ALNS algorithm, operators specialized at our problem may be researched and the performance of different operators may be evaluated. In this work, experiments were conducted instances with small size due to time limitation. The behavior of our model on large random instances may be investigated in the future.

# 8 Conclusion

This work introduced a new Vehicle Routing Problem, the Vehicle Routing Problem with Pickup and Delivery, Time Windows and Priority. It extended the classical VRP-PDTW with additional priority constraints by assigning membership status for each request and allowing time window violations for non-members' requests. Our problem is common in real-life when the customers of transportation services require different levels of service quality. An linear programming model was introduced for this problem.

A modified version of ALNS based on Ropke and Pisinger (2006) was proposed based on our model. We adapted the Shaw Removal heuristic to take the difference of membership status into consideration. The proposed ALNS was proved to be robust and efficient when applied to our model on instances with small size.

During the experiments, a set of instances with different member percentages were introduced based on Li and Lim (2003). Experiments were conducted with different sets of objective weights used in our model. The influence of objective weights and instance type were discussed. Our model is sensitive to the objective weights change when the fraction of weights between distance and delay are neither too small nor too larger. Delay increases and distance decreases as the fraction gets larger, and delay responds to weights change at a faster speed. The influence of objective weights is more prominent when the problem size is larger or our model is applied under the condition that the fraction of orders by non-member customers is larger.

# References


Baldacci, R., Bartolini, E., and Mingozzi, A. (2011). An exact algorithm for the pickup and delivery problem with time windows. *Operations research*, 59(2):414–426.

Barceló, J., Grzybowska, H., and Orozco, J. A. (2017). *City Logistics*, pages 1–44. Springer International Publishing, Cham.

Bent, R. and Van Hentenryck, P. (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 33(4):875–893.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Cattaruzza, D., Absi, N., Feillet, D., and González-Feliu, J. (2017). Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51–79.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

Desrosiers, J., Dumas, Y., and Soumis, F. (1986). A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6(3-4):301–325.

Dueck, G. and Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of computational physics*, 90(1):161–175.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European journal of operational research*, 54(1):7–22.

Gendreau, M., Guertin, F., Potvin, J.-Y., and Séguin, R. (2006). Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004). Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Science*, 38(2):245–255.

Li, H. and Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 12(02):173–186.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269.

Lu, Q. and Dessouky, M. (2004). An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4):503–514.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377–386.

Montané, F. A. T. and Galvão, R. D. (2002). Vehicle routing problems with simultaneous pick-up and delivery service. *Opsearch*, 39(1):19–33.

Nanry, W. P. and Barnes, J. W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435.

Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154.

Psaraftis, H. N. (1983). k-interchange procedures for local search in a precedence-constrained routing problem. *European Journal of Operational Research*, 13(4):391–402.

Ropke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Ruland, K. and Rodin, E. (1997). The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & mathematics with applications*, 33(12):1–13.

Santini, A., Ropke, S., and Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24(5):783–815.

Sexton, T. R. and Bodin, L. D. (1985). Optimizing single vehicle many-to-many operations with desired delivery times: Ii. routing. *Transportation Science*, 19(4):411–435.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer.

Shen, Y., Potvin, J.-Y., Rousseau, J.-M., and Roy, S. (1995). A computer assistant for vehicle dispatching with learning capabilities. *Annals of operations research*, 61(1):189–211.

SINTEF (2008). Li & lim benchmark. `https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark`.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265.

Toth, P. and Vigo, D. (2002). *The vehicle routing problem*. SIAM.

Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Van der Bruggen, L., Lenstra, J. K., and Schuur, P. (1993). Variable-depth search for the single-vehicle pickup and delivery problem with time windows. *Transportation Science*, 27(3):298–311.

Wilson, N. H., Weissberg, R. W., and Hauser, J. (1976). Advanced dial-a-ride algorithms research project. Technical report.

# A The ALNS framework applied on Li and Lim benchmark instances

Table 7: The ALNS framework on benchmark instances

| Test Case | Vehicle Number | Vehicle Number Gap | Objective | Optimal Objective | Objective Gap |
|---|---|---|---|---|---|
| LC101 | 10 | 0.000000 | 828.936867 | 828.94 | 0.000004 |
| LC102 | 10 | 0.000000 | 828.936867 | 828.94 | 0.000004 |
| **LC103** | **10** | **0.111111** | **827.864699** | **1035.35** | **-0.200401** |
| **LC104** | **10** | **0.111111** | **818.599984** | **860.01** | **-0.048151** |
| LC105 | 10 | 0.000000 | 828.936867 | 828.94 | 0.000004 |
| LC106 | 10 | 0.000000 | 828.936867 | 828.94 | 0.000004 |
| LC107 | 10 | 0.000000 | 828.936867 | 828.94 | 0.000004 |
| LC108 | 10 | 0.000000 | 826.439205 | 826.44 | 0.000001 |
| **LC109** | **10** | **0.111111** | **827.816613** | **1000.60** | **-0.172680** |
| LC201 | 3 | 0.000000 | 591.556557 | 591.56 | 0.000006 |
| LC202 | 3 | 0.000000 | 591.556557 | 591.56 | 0.000006 |
| LC203 | 3 | 0.000000 | 591.173443 | 591.17 | 0.000006 |
| LC204 | 3 | 0.000000 | 590.598746 | 590.60 | 0.000002 |
| LC205 | 3 | 0.000000 | 588.875963 | 588.88 | 0.000007 |
| LC206 | 3 | 0.000000 | 588.492849 | 588.49 | 0.000005 |
| LC207 | 3 | 0.000000 | 588.286321 | 588.29 | 0.000006 |
| LC208 | 3 | 0.000000 | 588.323801 | 588.32 | 0.000006 |
| LR101 | 19 | 0.000000 | 1650.799240 | 1650.80 | 0.000000 |
| LR102 | 17 | 0.000000 | 1487.570433 | 1487.57 | 0.000000 |
| LR103 | 13 | 0.000000 | 1292.675510 | 1292.68 | 0.000003 |
| LR104 | 9 | 0.000000 | 1013.389290 | 1013.39 | 0.000001 |
| LR105 | 14 | 0.000000 | 1377.111018 | 1377.11 | 0.000001 |
| LR106 | 12 | 0.000000 | 1252.616654 | 1252.62 | 0.000003 |
| LR107 | 10 | 0.000000 | 1111.313176 | 1111.31 | 0.000003 |
| LR108 | 9 | 0.000000 | 968.966019 | 968.97 | 0.000004 |
| LR109 | 11 | 0.000000 | 1208.964779 | 1208.96 | 0.000004 |
| LR110 | 10 | 0.000000 | 1159.348406 | 1159.35 | 0.000001 |
| LR111 | 10 | 0.000000 | 1108.900730 | 1108.90 | 0.000001 |
| **LR112** | **10** | **0.111111** | **1027.118571** | **1003.77** | **-0.023261** |
| LR201 | 4 | 0.000000 | 1253.233969 | 1253.23 | 0.000003 |
| ***LR202** | **4** | **0.333333** | **1250.089987** | **1197.67** | **0.043768** |
| LR203 | 3 | 0.000000 | 949.396408 | 949.40 | 0.000004 |

| | | | | | |
|---|---|---|---|---|---|
| LR204 | 2 | 0.000000 | 849.050423 | 849.05 | 0.000000 |
| LR205 | 3 | 0.000000 | 1054.018630 | 1054.02 | 0.000001 |
| LR206 | 3 | 0.000000 | 931.625436 | 931.63 | 0.000005 |
| LR207 | 2 | 0.000000 | 903.055621 | 903.06 | 0.000005 |
| LR208 | 2 | 0.000000 | 734.848036 | 734.85 | 0.000003 |
| LR209 | 3 | 0.000000 | 930.585749 | 930.59 | 0.000005 |
| LR210 | 3 | 0.000000 | 964.223563 | 964.22 | 0.000004 |
| **LR211** | **3** | **0.500000** | **884.294002** | **911.52** | **-0.029869** |
| **LRC101** | **15** | **0.071429** | **1703.214515** | **1708.80** | **-0.003269** |
| LRC102 | 12 | 0.000000 | 1558.069346 | 1558.07 | 0.000000 |
| LRC103 | 11 | 0.000000 | 1258.737118 | 1258.74 | 0.000002 |
| LRC104 | 10 | 0.000000 | 1128.401171 | 1128.40 | 0.000001 |
| LRC105 | 13 | 0.000000 | 1637.624391 | 1637.62 | 0.000003 |
| ***LRC106** | **12** | **0.090909** | **1440.442512** | **1424.73** | **0.011028** |
| LRC107 | 11 | 0.000000 | 1230.144845 | 1230.14 | 0.000004 |
| LRC108 | 10 | 0.000000 | 1147.425288 | 1147.43 | 0.000004 |
| LRC201 | 4 | 0.000000 | 1406.940088 | 1406.94 | 0.000000 |
| ***LRC202** | **4** | **0.333333** | **1390.564587** | **1374.27** | **0.011857** |
| LRC203 | 3 | 0.000000 | 1089.067688 | 1089.07 | 0.000002 |
| LRC204 | 3 | 0.000000 | 818.663061 | 818.66 | 0.000004 |
| LRC205 | 4 | 0.000000 | 1302.198474 | 1302.20 | 0.000001 |
| LRC206 | 3 | 0.000000 | 1159.033183 | 1159.03 | 0.000003 |
| LRC207 | 3 | 0.000000 | 1062.048312 | 1062.05 | 0.000002 |
| LRC208 | 3 | 0.000000 | 852.757596 | 852.76 | 0.000003 |

Vehicle Number and Objective reports the results of our ALNS. Optimal Objective reports the best objective value found in the literature, which is available at SINTEF (2008). Gap is defined as the difference of result of the ALNS and the optimal divided by the optimal. Instances marked bold are instances with objective value different from reported optimal, instances with additional * are with higher objective.

# B Experiment results for different objective weights

Table 8: Average objective value for each combination of instance class and member percentage

| Instance Class | Member Percentage | $(\alpha, \beta)$ | Vehicle Number | Objective | Distance | Delay |
|---|---|---|---|---|---|---|
| LR1 (size 100) | 0.3 | (1,3) | 14.416667 | 1382.426095 | 1368.578300 | 4.615932 |
| | | (1,1) | 14.150000 | 1366.312282 | 1346.026694 | 20.285588 |
| | | (3,1) | 13.083333 | 4000.562436 | 1296.304613 | 111.648595 |
| | 0.5 | (1,3) | 14.500000 | 1385.359897 | 1378.502490 | 2.285803 |
| | | (1,1) | 14.333333 | 1374.141177 | 1359.212363 | 14.928814 |
| | | (3,1) | 13.916667 | 4058.661753 | 1328.462978 | 73.272818 |
| | 0.7 | (1,3) | 14.883333 | 1394.816304 | 1391.179319 | 1.212328 |
| | | (1,1) | 14.616667 | 1386.462528 | 1378.278294 | 8.184234 |
| | | (3,1) | 14.200000 | 4115.416218 | 1355.582194 | 48.669637 |
| LR2 (size 100) | 0.3 | (1,3) | 4.745455 | 1207.838200 | 1206.088115 | 0.583361 |
| | | (1,1) | 4.581818 | 1202.738156 | 1192.610871 | 10.127286 |
| | | (3,1) | 4.345455 | 3522.206029 | 1142.554802 | 94.541622 |
| | 0.5 | (1,3) | 4.763636 | 1206.122019 | 1205.381935 | 0.246695 |
| | | (1,1) | 4.690909 | 1204.505669 | 1198.381530 | 6.124139 |
| | | (3,1) | 4.436364 | 3549.360859 | 1161.760065 | 64.080663 |
| | 0.7 | (1,3) | 4.872727 | 1206.943814 | 1206.606965 | 0.112283 |
| | | (1,1) | 4.800000 | 1209.016136 | 1206.780693 | 2.235443 |
| | | (3,1) | 4.781818 | 3604.264783 | 1193.124080 | 24.892543 |
| LR1 (size 200) | 0.3 | (1,3) | 16.300000 | 3711.700189 | 3678.530128 | 11.056687 |
| | | (1,1) | 16.000000 | 3679.628870 | 3630.912899 | 48.715971 |
| | | (3,1) | 14.280000 | 10855.865187 | 3466.380759 | 456.722910 |
| | 0.5 | (1,3) | 16.600000 | 3739.999692 | 3706.811843 | 11.062616 |
| | | (1,1) | 16.300000 | 3705.948768 | 3662.162321 | 43.786447 |
| | | (3,1) | 15.240000 | 10975.773449 | 3571.493397 | 261.293256 |
| | 0.7 | (1,3) | 16.700000 | 3766.894280 | 3743.415466 | 7.826271 |
| | | (1,1) | 16.380000 | 3743.551806 | 3706.137112 | 37.414694 |
| | | (3,1) | 15.500000 | 11101.616243 | 3637.725493 | 188.439765 |