The economics of same-day delivery by means of a variable neighborhood-based heuristic

Abstract

The same-day delivery routing problem is a generalization of the pickup and delivery vehicle routing problem where a heterogeneous fleet of vehicles is given, the pickup locations of customer requests are not predetermined and the time windows cannot be violated. Moreover, consolidation is considered at certain locations. In this work, a new insertion-based initial solution heuristic has been applied which is followed by a variable neighborhood search (VNS) algorithm. In the VNS algorithm, six neighborhoods are adapted to the new problem setting considering intra and inter route relocations, consolidation, vehicle upgrade and downgrade. A case study based on data and requirements from a big-box U.S. retailer has been conducted. Computational results and sensitivity analyses are presented to evaluate the variable neighborhood approach and to provide an economic assessment for the same-day delivery business model.

Keywords: Vehicle routing; Same-day delivery; Variable neighborhood search

1. Introduction

Nowadays, there is a growing interest in the same-day delivery problem in the logistics, retail and high-tech industries and implementations of same-day delivery have already been applied in the commercial sector as Amazon, Google, Walmart and Sears started to test such service in their micro-markets with a variety of assets. In general, the same-day delivery problem aims at delivering customer orders within 24 hours after the order placement. Each order can contain several requests or items which can be dispatched independently. For each request a customer can opt to either be delivered at a specified location (home delivery), or to be picked up at a depot (self-pickup). Same-day delivery routing is to minimize the transportation cost by scheduling requests' pickup and delivery into different routes obeying time windows. Several business strategies and requirements are possible in the same-day delivery routing context. First, there are trade-offs among different types of vehicles. In congested areas like Manhattan, a bicycle may be a better transportation mode due to low fixed cost, zero fuel cost and convenience while a van may be favored in rural areas due to its speed and economy of scale. In order to better utilize fleet resources and find ways to reduce daily cost, the same-day delivery routing model should capture this feature and optimize the mix of fleets in the routing solution. Second, when several stores can provide the same request, there is an option of selecting the best store in that route to reduce the delivery cost. This elicits another feature of the same-day delivery routing problem which is selective pickup. Third, there is a cost reduction opportunity when a consolidation point allows requests to be brought in from one route and then delivered in another route. Clearly, the traditional constraints of vehicle capacity and time window must be obeyed. In essence, the strategies and constraints are used to leverage strength and resources of the commercial business. And strategic decisions (e.g. asset allocation, service price, etc.) can be synthesized from the same-day delivery routing problem.

The same-day delivery routing problem is related to the generalized E-grocery delivery routing problem (EDRP) faced in online grocery (Emeç et al., 2016). Yanki et al. (2014) and Emeç et al. (2016) design heuristics for solving EDRP, which is a capacitated VRP with multiple pickups, single delivery and time windows. In EDRP, a given fleet of vehicles originate from a single depot and visit assigned vendor locations to pick up the requests before delivering to customers. For some of the requests, the pickup location has not been determined and could be sourced from multiple vendors. The delivery of requests takes place only after the entire set of requests are collected and combined with the requests loaded at the depot (Emeç et al., 2016). Its objective is minimizing the total distance traveled or cost incurred. Similar to EDRP, the same-day delivery routing problem addressed in this work concentrates on assigning requests to a set of routes which complete the pickup and delivery tasks within one day. Since requests from the same order can be picked up in different locations (either at a store or depot), we decompose each order into its requests and consider requests' assignment. In response to the business strategies and constraints illustrated before, the same-day delivery routing problem includes new features in addition to or distinct from the EDRP. First, a heterogeneous fleet of vehicles that differ in speed, capacity and cost rate are considered to leverage their advantages (as opposed to a single fleet considered in EDRP). We do not consider traffic congestion so vehicle speeds are static. Different from other works, we do not consider vehicle fixed cost since they can be armotized. The variable cost rates are calculated based on fuel and labor costs which are related to the distance and time traveled. Second, a single consolidation point, which is the depot, is considered. The EDRP does not consider consolidation. Third, we allow multiple shipments of a request, i.e, the delivery of requests can happen without having the entire order collected. The EDRP considers multiple pickup and delivery simultaneously in the route assignment with the time window constraint but still in its vanilla version does not include the above properties. Our setting is dictated and based on our industrial partner which is a big box US retailer. For example, multiple fleets correspond to the possitiblity of having several modes of transporation (car, bike, van, etc), and requests from a single order can be delivered unatended. The same-day delivery routing problem studied herein is a tactical or strategic study and thus not a real-time operations problem. Instead of generating real-time orders when the vehicles are en route, we consider orders processed in batch before routes start. For this reason we have cut-off times during a day and orders placed before a cut-off time are required to be fulfilled by a certain later time (in our case there are 3 different cut-off times: early morning, early afternoon, and late afternoon). .

The methodology deployed in this work includes a new insertion-based initial solution heuristic and a VNS algorithm. Since the insertion method is effective in constructing static routes, we start by constructing an initial solution with a modified insertion method. For the VNS algorithm, we construct six distinct neighborhoods which capture the features of the same-day delivery problem to explore improving solutions. Based on empirical data, we systematically find the most effective way to apply the neighborhoods leading to the best solution.

With data and problem setting coming from our business partner, in the case study section, we define two types of problems: the default same-day delivery problem (i.e. no limits on fleet resources) and the fleet constrained problem. In the case study, requests are assumed to have four kinds of sources: depot, self-owned stores, local merchants that prepare requests ahead of time and local stores requiring shopping (independent local stores with no prearranged agreement). A different kind of a store has its own pickup time window policies. Orders for home delivery must be delivered approximately 12 hours after the placement while self-pickup orders roughly 8 hours after placement. This rule is aggregated in the tactical or strategic study into morning, early afternoon and late afternoon. For example, orders placed before 7:00 am have to be delivered before 8:00 pm (morning delivery). Cost sensitivities to fleet resources, time window policy and service time are tested. In addition to obvious facts such as the car being the most desired fleet, we discover from the case study that: (1) in the default same-day delivery problem, approximately 5% of total requests can be consolidated and the consolidation heuristic in the VNS algorithm contributes a 5.5% total cost reduction; (2) in the default same-day delivery problem, the fleet utilization rate is more than 70%; (3) substantially lowering the hourly pay for bicycle operations does not encourage this fleet to be used; (4) increasing van's cost efficiency to a certain point makes it economical for use; (5) in the fleet constrained problem, van is better in replacing car than bicycle due to a potential lack of cars; (6) reducing shopping time or depot handling time does not reduce total cost significantly; (7) moving the morning home delivery time 8 hours earlier for requests requiring pickup from either the retailer's own stores or local merchants increases the total cost by 10%; (8) pushing forward the noon home delivery time 2 hours earlier for requests requiring pickup at the retailer's own stores or local merchants increases the total cost by 5%; (9) in a comparable size problem , the total cost in a high population density area is slightly lower than in a medium population density area.

The most important contribution of this paper is to present a new VNS algorithm for solving the same-day delivery routing problem. Besides the normal features in EDRP, the algorithm captures the trade-off between different types of vehicles and opportunities for consolidation under the time window and the vehicle capacity constraint. We combine these features together and provide effective approaches. Another contribution is the case study based on real world scenarios that considers pilot strategies of a big-box retailer. It validates the effectiveness of the developed strategies.

The rest of the paper is organized as follows. Section 2 provides a literature review for the classic VRP, PDPTW, pickup and delivery problem with transfer (PDPT), mixed fleet VRP and VNS. Section 3 formally states the problem and discusses the solution methodology. Section 4 is the case study part which includes a data discussion, computational results and sensitivity analyses.

2. Literature Review

The classic VRP has been extensively studied over past decades and can be reviewed in Toth and Vigo (2001). With regard to the mixed fleet VRP, both meta-heuristics and mathematical programing methods have been used. Additional information and reviews can be found in Salhi and Rand (1993), Osman and Salhi (1996) and Lee et al. (2006). PDPTW has also received attention in the past decades and we refer to Berbeglia et al. (2007) for further reading. Mladenovi ć and Hansen (1997) developed VNS to solve combinatorial and global optimization problems.

Contrary to other meta-heuristics based on local search methods, VNS does not follow a fitted trajectory but explores increasingly distant neighborhoods of the current incumbent solution and jumps to a new solution if an improvement has been made (Hansen and Mladenović, 2001). It has been widely used in solving the classic VRP and its extensions.

The traditional selective pickup and delivery problem has a many-to-many scheme, where each node serves as either a source (pickup) or a destination (delivery) of commodities and the commodities collected from pickup nodes can supply any delivery node (Berbeglia et al., 2007). This type of selective pickup commonly appears in reverse logistics. For example, in the soft drink industry, vehicles have to pick up and bring back empty containers while delivering full containers to customers. Due to limited capacity, decisions have to be made about which pickup location is to be visited. Classic construction and improvement heuristics, as well as a tabu search heuristic were developed and tested on a number of instances by Gribkovskaia et al. (2008) on a single vehicle pickup and delivery problem with common selective pickups. Moreover, Gutiérrez-Jarpa et al. (2010) solved a VRP with deliveries, selective pickups and time windows by using a branch and price algorithm where the algorithm was also capable of handling five variants concerning single (all delivery demands originate at the depot and all pickup demands are destined to the depot) or combined demands (the same customer may have a pickup and a delivery), single or mixed routes, etc. The term "selective pickup" has another meaning in our same-day delivery routing problem. First, one request can be picked up at different locations. Second, pickup and delivery locations are disjoint which means that at a customer location there is no pickup. This type of selective pickup is also called multiple pickup and has been considered in some previous works. Yanik et al. (2014) developed a hybrid approach to solve the multiple-pickup single delivery vehicle routing problem in the presence of multiple vendors. A genetic algorithm was applied to assign vendors to customers in their work. A modified savings algorithm was used in the routing part. Emeç et al. (2016) introduced six vendor selection mechanisms into their adatptive large neighborhood search (ALNS) heuristic. The vendor selection process is generally determined by the distance between pickup and delivery locations, or based on each pickup's historical performance. In our work, we greedily search the best pickup location which brings the lowest cost increase to the solution.

As a recent extension, the pickup and delivery problem with consolidation has not been studied extensively. This concept is first used in the pickup and delivery problem of Shang and Cuff (1996) as "transfer." In general, studies of this aspect can be categorized by the number of consolidation points: multiple or single. For multiple consolidation points, in the study of Shang and Cuff (1996), any point, any item and any two vehicles can be part of the consolidation process. They applied a pickup and delivery heuristic with a built-in mini-route construction heuristic to realize the consolidation. Thangiah et al. (2007) proposed a heuristic method with the algorithm adapted from Shang and Cuff (1996). Three local optimization methods were identified: an overlap heuristic, exchange heuristic and insert unassigned heuristic. Mitrović-Minić and Laporte (2006) quantified the benefits of consolidation with a two-phase heuristic (construction and improvement) for the static pickup and delivery problem where consolidations are predetermined. Several methods of pre-locating consolidation points are used in their work (e.g. randomly selected and determined by geographical rules). For a single consolidation point, Qu and Bard (2012) evaluated their greedy randomized adaptive search procedure and a further improvement strategy in adaptive large neighborhood search in instances with 25 requests and 1 consolidation point. Petersen and Ropke (2011) used the large neighborhood search method in solving a similar pickup and delivery problem with 1 consolidation point corresponding to the depot where all vehicle routes start and end. In our problem setting, a single consolidation point is considered.

The mixed fleet VRP is known as the heterogeneous VRP (HVRP). Studies on HVRP can be traced back to 30 years ago when Golden et al. (1984) explored the giant tour algorithms. Generally, the family of HVRP considers a fleet of vechicles with different capacity, fixed cost and variable cost in addition to the classic VRP aspects. As identified by Koç et al. (2016), HVRPs have two major variations: the fleet size and mix vehicle routing problem (FSM) introduced by Golden et al. (1984) and the heterogeneous fixed fleet vehicle routing problem (HF) introduced by Taillard (1999). The first variation solves the problem with unlimited number of vehicles while the later one models a fixed number of vehicles. Both variations can be solved by exact algorithms (Baldacci, et al. 2009; Pessoa, et al. 2009; Choice and Tcha, 2007), continuous approximations models (Jabali, et al. 2012) or heuristics (Taillard, 1999; Gendreau, et al. 1999; Wassan and Osman, 2002). For details of different methods, we refer to Koç et al. (2016). The algorithm we proposed solves a heterogenous fleet with unlimited number of vehicles. In the sensitivity analyses section of the case study, a fleet selection algorithm is designed to handle limited vehicles. In the VNS section, we provide three hybrid neighborhoods combining vehicle assignment and routing reconstructuring.

To the best of our knowledge, no studies have been focused on the pickup and delivery problem with consolidation defined in our same-day delivery routing problem. On the other hand, it is the first time of combining mixed fleet, pickup and delivery, selective pickup and consolidation aspects in a problem setting with two constraints: time window and vehicle capacity. Furthermore, VNS has never been used in this complex same-day delivery routing problem.

3. Model and Methodologies

3.1 Problem Setting

The same-day delivery routing problem can be defined mathematically as follows. There is a set Q of requests to be picked up and delivered. Let P be the set of all predetermined pickup locations or in network parlance vertices. Each request $i \in Q$ has to be picked up from set $P_{(i)} \subseteq P$ (i.e. $P_{(i)}$ can be the set of all Target stores at different locations). We denote $p_{(i)} \in P_{(i)}$ as its selected pickup location (decision to be made) and we denote the delivery vertex as $d_{(i)}$. Let $G = (V, E)$ be a complete graph with vertex set $V = P \cup D \cup \{0\}$ where pickup vertex set $P = \bigcup_i P_{(i)}$, delivery vertex set $D = \bigcup_i d_{(i)}$ and $\{0\}$ is the depot. Set $E = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$ is the edge set. Also, we define a fleet of K vehicles $F = \{f_1, \dots, f_K\}$ with weight capacity $\{w_{(f_1)}, w_{(f_2)}, \dots, w_{(f_K)}\}$ and volume capacity $\{v_{(f_1)}, v_{(f_2)}, \dots, v_{(f_K)}\}$. Some of these vehicles are often going to be of the same vehicle type. We define solution set R as a collection of routes. A route is defined in the way that vehicles start from the depot loaded with requests sourced from the depot, next visit pickup or delivery locations, and finally return to the source depot (possibly still loaded with some items for the purpose of consolidation). Both Q and P have hard pickup or delivery time windows. In terms of Q, for each request $i \in Q$, the time window is $[e_{(i)}, l_{(i)}]$, where $l_{(i)}$ is the delivery due time and

$e_{(i)}$ represents the earliest delivery time. In the same-day delivery routing context, $e_{(i)}$ indicates that the vehicle cannot visit the customer before the order placement happens and $l_{(i)}$ is a business policy. As this is not a real-time operations problem, requests in Q are processed in multiple batches $\{Q_0, \dots, Q_B\}$ with requests in the same batch having the same earliest delivery time. As for each store $j \in P$, $[e_{(j)}, l_{(j)}]$ represents merchants' operating timetable. In addition, we also consider other business rules that may alter the time window at pick up locations. For example, after the request is placed, some merchants require a lead time to prepare this request. So the vehicle cannot pickup the request before it is prepared. Another constraint is the capacity of the vehicle that is measured in two dimensions here: weight and cube. Either one is not allowed to be violated. Table 1 in the Appendix summarizes the notation. The total cost of routes is measured in terms of the operating cost. Although some studies take into account fixed cost, i.e., the total cost incorporates a fixed cost component depending on the vehicle type chosen plus the cost proportional to the travel distance (Lee et al., 2008). In our work, we only consider the variable cost since the fixed cost can be easily considered by an amortizing method if required. The total cost includes the labor and fuel costs. The labor cost is calculated by the travel time and hourly rate while the fuel cost is computed based on the travel distance. The "en route time" here also includes the pickup time spent in stores, store shopping times and depot handling times. Specifically, the store shopping time is estimated by considering steps in the shopping process including shopping, at the cashier and loading time.

To illustrate the same-day delivery problem, an example is provided in Figure 1. The square represents the depot, the octagons represent the pickup stores, and the circles represent the delivery locations. Denote q_i for request i and ps_j for store j. We start with five requests $Q = \{q_1, \dots, q_5\}$ and $P_{(q_1)} = \{ps_1, ps_2\}, P_{(q_2)} = \{ps_3, ps_4\}, P_{(q_3)} = \{ps_5\}, P_{(q_4)} = \{ps_1, ps_2\}$. We have q_5 loaded at the depot. The delivery locations are $D = \{d_1, \dots, d_5\}$. In Figure 1, we use $load = \{\}$ along the edge indicating the request the vehicle is carrying and t to indicate the time a vehicle visits a vertex. Assume we have a homogenous fleet. In the displayed solution, we have two routes. The first route is $\{0\} \to \mathrm{ps}_1 \to d_1 \to ps_5 \to d_3 \to ps_3 \to \{0\}$. The time the vehicle visits each node is $\{\mathrm{t}_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. The second route is $\{0\} \to \mathrm{d}_4 \to d_5 \to d_2 \to \{0\}$. The time the vehicle visits each node in $\{\mathrm{t}_8, t_9, t_{10}, t_{11}, t_{12}\}$. As the graph shows, the first route brings back q_2 and q_4, then the second route delivers them. In the figure, t_7 is the back time of the first route and t_8 is the start time of the second route. We have $t_7 < t_8$. For q_2 and q_4, we have $e_{(q2)} \le t_{11} \le l_{(q2)}$ and $e_{(q4)} \le t_9 \le l_{(q4)}$. For visiting pickup vertex, we ensure the time window constraint. For example, $e_{(ps1)} \le t_2 \le l_{(ps1)}$. For visiting delivery vertex, we guarantee the time window as well. For example, $e_{(q1)} \le t_3 \le l_{(q1)}$ ensures the time the vehicle visits d_1 is larger than the earliest time to deliver $q1$ but less than the latest time to deliver this request. In the solution, we have the pickup location assignment as: $p_{(q_1)} = ps_1, p_{(q_2)} = ps_3$ and $p_{(q_4)} = ps_1$.

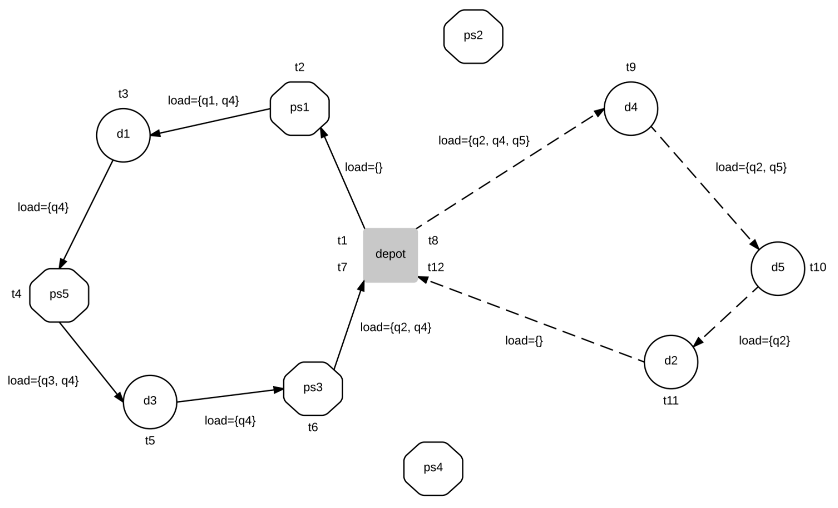

Figure 1. An illustrative example

Comment [J1]: Inside paper, change index

3.2 Initial Solution

The algorithm of constructing an initial solution is based on the least cost insertion method and it aims at finding the insertion position with the least extra cost. Nanry and Barnes (2000) present a simple predecessor-successor pair insertion algorithm to obtain a feasible solution and then a reactive tabu search for PDPTW. In their method, pickup and delivery nodes are paired, and then the least cost pair insertion method is applied. In our work, the construction of the initial solution is based on the method of least cost pair insertion but it is more complex than the predecessor-successor pair insertion algorithm for three reasons: (1) for one request i, its pickup has not been determined and there are several $[p_{(i)}, i]$ pairs with $p_{(i)} \in P_{(i)}$; (2) in some situations, request i and its pickup $p_{(i)}$ may not be in the same route as consolidation is allowable; (3) a fleet upgrade heuristic is integrated to provide an alternative when pair insertion fails. In the remainder of this section, we provide details accounting for these complexities.

We are given a set of routes not necessarily fulfilling all requests (indeed, in the initial solution construction algorithm we assume that the requests in Q are not yet covered). First, all requests $i \in Q$ are sorted by their late delivery window $l_{(i)}$ from early to late, and the order of pair insertion follows this sorted Q. This approach is based on the intuition that requests with tighter windows should be served first. On the other hand, this approach essentially provides consolidation opportunities because requests with larger window lengths are inserted later and when they cannot be inserted, they may be consolidated from two routes due to their more flexible window. Meanwhile, all vehicles $f \in F$ are sorted by their economic efficiency ratio $ef_{(f)}$ defined as cost per mile (the higher efficiency has a lower ratio) and the available vehicle with highest economic efficiency is initially chosen when constructing a new route. A less efficient vehicle is considered only when all more efficient vehicles have already been dispatched. Details about this algorithm are in Algorithm 5. Second, request i with the earliest $l_{(i)}$ is inserted into the first route (newly created routes are inserted at the beginning of the list of all routes). In this process, as each request i has a pickup store list $P_{(i)}$ which is a subset of P, each $p_{(i)} \in P_{(i)}$ is evaluated and the least cost pickup location is decided. The algorithm then attempts to insert subsequent requests and their best pickup locations into the route until no request can be inserted into this route without violating time window and vehicle capacity constraints.

Two reasons lead to insertion failure: (1) inserting one request i and its $p_{(i)} \in P_{(i)}$ always causes the vehicle overloads at some vertex; (2) inserting one request i and its $p_{(i)} \in P_{(i)}$ always violates the time window constraints. If the first reason occurs, it may be profitable to first replace the vehicle of the current route with a larger but less efficient vehicle, then reinsert the failed request. The rationale behind this vehicle replacement approach is that to construct a new route may cost more compared to leveraging the existing route since (1) a new route uses a vehicle resource; (2) if the replacement is successful, the extra capacity may further reduce cost when intra-route relocation is applied. Hence, before constructing a new route with the rest of the requests, it is reasonable to assess whether replacing the current route with a larger vehicle may bring any economic benefits. A simple diagram presents this process in Figure 2. For example, a^+ represents the pickup vertex of request a while a^- is the delivery vertex. It shows that pair d^+ and d^- (in bold) cannot be inserted to route 1 (denoted as r_1) with vehicle 1 (denoted as f_1) due to the overloading problem. One option of a successful insertion may be introduced by upgrading $r_1's$ vehicle with vehicle 2 (denoted as f_2) where $v_{(f_2)} > v_{(f_1)}$ and $w_{(f_2)} > w_{(f_1)}$. Another option is to start a new route with the most efficient and currently available vehicle, for example f_1. The costs of both options are then compared and the best one is selected. For each option, every possible pickup location in the selective list is considered.

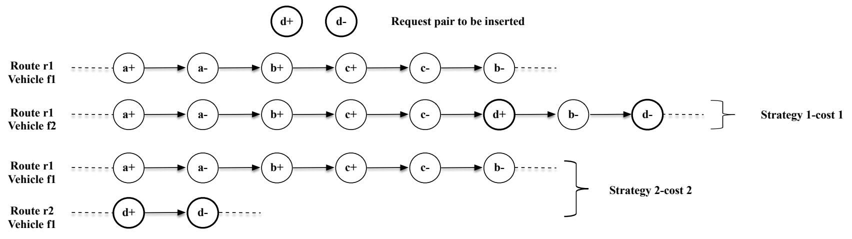

Figure 2. Vehicle upgrade in initial solution

The consolidation heuristic is described as follow. After at least two routes, for example r_1 and r_2 have been built, there is an opportunity to consolidate the next request via the consolidation point. Suppose their start and return times are denoted by $s_{(r_1)}, s_{(r_2)}, rb_{(r_1)}, rb_{(r_2)}$, respectively and we are inserting request i paired with each possible pickup from the list $P_{(i)}$. Then consolidation can be applied in this situation by picking up the request in r_1, then dropping it at the consolidation point, and another vehicle delivers it in r_2, only if condition $rb_{(r_1)} < s_{(r_2)}$ holds and some $p_{(i)} \in P_{(i)}$ has already been visited in r_1. The advantage here is to utilize visited pickup locations in an existing route and avoid revisiting the same vertex again. Figure 3 presents this situation and conditions to be held. The top two routes are the current routes while the bottom two routes are the updated routes after insertion. Regarding the order of applying the vehicle upgrade or consolidation heuristic, it is hard or computationally expensive to measure which approach to apply first. In the case study, we first apply the vehicle upgrade heuristic and then the consolidation heuristic as we observe consolidation is more likely to occur. Algorithm 1 provides details about constructing the initial solution.

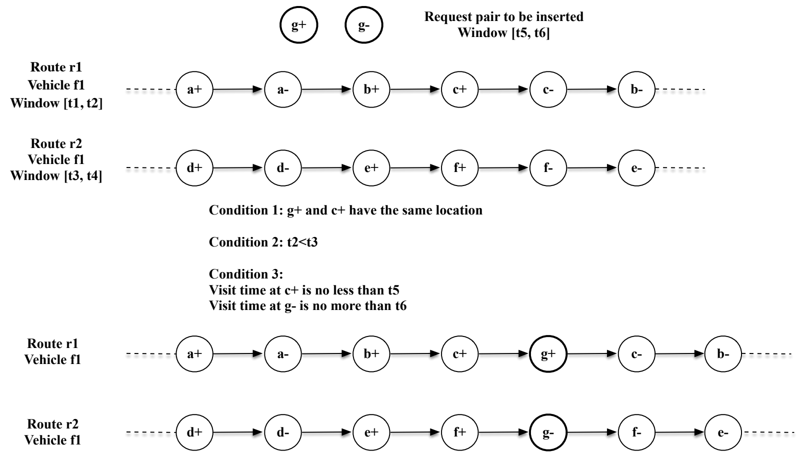

Figure 3. Consolidation in initial solution

```
Algorithm 1: Initial Solution Construction
Input:
Priority queue $Q$, Set $F$
Initialize set $R = \emptyset$
Sort $Q$ by $l_{(i)}$ in the ascending order
Sort $F$ by $ef_{(f)}$ in the ascending order
While $Q$ is not empty do
    Pop front request $j$ from $Q$
    Initialize a new route $r$ with request $j$ and add $r$ to $R$, apply vehicle selection method to select
    $f \in F$
    While there are unprocessed requests in $Q$
        Pop front request $i$ from $Q$
        Apply the insertion method for request $i$ to route $r$
        If insertion is unsuccessful
            Push back request $i$ to the end of $Q$
        End
    End
    While there are unprocessed requests in $Q$
        Pop front request $i$ from $Q$
        Apply the vehicle upgrade method on route $r$
        If upgrade is unsuccessful
            Push back request $i$ to the end of $Q$
        End
    End while
    While there are unprocessed requests in $Q$
        Pop front request $i$ from $Q$
        Apply the consolidation method
        If consolidation is unsuccessful
            Push back request $i$ to the end of $Q$
        End
    End while
End while
```

3.3 Variable Neighborhood Search

After constructing an initial solution, six neighborhoods: intra-route relocation, inter-route relocation, consolidation relocation, vehicle upgrade-1, vehicle upgrade-2 and vehicle downgrade are selected to improve the solution. First, these neighborhoods are explained and then the overall VNS algorithm is presented. In this section we assume that there are no constraints on the fleet size.

3.3.1 Neighborhoods

Intra-route relocation. This heuristic is to shift request i and $p_{(i)} \in P_{(i)}$ within the current route while keeping the feasibility of the solution. Different from traditional intra-route relocation (Nanry and Barnes, 2000), this heuristic includes selective pickup. When relocating pair $[p_{(i)}, i]$, each $p_{(i)} \in P_{(i)}$ is attempted to make sure the best pickup location is chosen. A relocating example is depicted in Figure 4. In this example, d^+ and d^- are removed from the original route and we present two options to insert d^- when d^+ has already been inserted. It starts by inserting $p_{(i)}$ first at every possible position and then moving the delivery vertex to a feasible position after the pickup position. The relocation process can be considered as two steps: (1) for the incumbent route r, repeat $p_{(i)} \in P_{(i)}$ and find the best insertion position for pair $[p_{(i)}, i]$ and store the cost savings into a list; (2) find the lowest cost value in this list and the best pickup $p_{(i)}$.

Under the time window and capacity constraints, the computational cost comes from reevaluating the time window constraint at each vertex. For a large-scale problem, it is not efficient to attempt every request in the route. So, we introduce parameter $\alpha_1 \in [0,1]$ capturing that only α_1 percentage of requests can be moved. Later in the case study, we present a method to tune this parameter. Once α_1 has been determined, it is applied to all selected routes.

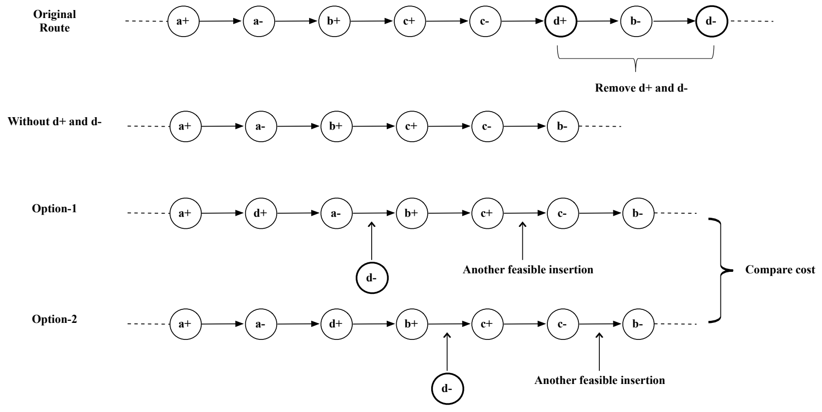

Figure 4. Pair relocation operator in intra-route neighborhood

Inter-route relocation. This heuristic is to swap two requests between two routes. Suppose we have two requests i and j. A traditional pair swap is to put the successor of pair $[p_{(i)}, i]$ into the position of the successor of pair $[p_{(j)}, j]$ and the predecessor of pair $[p_{(i)}, i]$ into the position of the predecessor of pair $[p_{(j)}, j]$ and vice versa. It is helpful for the solution to escape from a local optimum as it alters the structure of the current solution. In our algorithm, this inter-route relocation heuristic becomes more complicated than a simple swap and it is more similar to reinsert i and j to each other's route. It takes out requests i and j from their routes r_1 and r_2, respectively. Then request i is reinserted into r_2 while request j is reinserted into r_1. In the reinsertion process, we attempt to evaluate other possible pickup locations in $P_{(i)}$ or $P_{(j)}$ as the current optimal pickup location may not be favored in the new route. Although it takes more computational time than simply swapping the two pairs, it performs better in jumping out of a local optimum. Because swapping is sensitive to the time window and capacity constraints in our problem, most of the trials may fail because they cannot meet some of the constraints. In execution, this heuristic randomly selects an α_2 percent of pairs from the pair set $\{(i, j)\}$ where i belongs to the request list of r_1 and j belongs to the request list of r_2. Each selected pair (i, j) is then considered for swapping.

However, additional complexity occurs when the request to be swapped has been consolidated. For example, i has been consolidated and it is picked up in route r_3 instead of r_1. When this happens, this pair is not to be swapped as it involves changing the structure of r_3. In summary, successful relocation conditions in this heuristic are: (1) both requests can be inserted into the other route without violating the time window and vehicle capacity constraints; (2) after the relocation, $[c_{(r_1)} + c_{(r_2)} > c_{(r_1')} + c_{(r_2')}]$ holds, where r_1' and r_2' are the updated routes.

Consolidation relocation. When constructing the initial solution, consolidation has already been considered. There are two reasons for reconsidering consolidation: (1) the heuristic constructing the initial solution only considers those requests that cannot be inserted during the pair insertion step; this ignores all remaining requests; (2) after a feasible initial solution has been generated, there are more routes than in the construction process. Similarly as before, this consolidation relocation heuristic aims at shaking the structure of the current solution by removing one request i and its pickup in $p_{(i)}$ from their current route and inserting them into different routes if and only if the total cost can be reduced. Every route pair in the current solution is considered for consolidation and the most beneficial one is chosen. We start this process by randomly selecting α_3 percent of unconsolidated requests. The difference between this heuristic and the one in the initial solution is that the initial heuristic only finds insertions that can leverage the visited vertex. However, in this heuristic, a successful consolidation happens if cost reduction exists and the algorithm selects the consolidation yielding the largest cost reduction. Algorithm 2 outlines this consolidation heuristic.

```
Algorithm 2: Consolidation Relocation
Input:
Set $R$
Initialize find route pairs $(r_m, r_n) \in R \times R$ where $rb_{(r_m)} < s_{(r_n)}$ and put these pairs into set $M'$
For each $r \in R$
  Let $Q$ be the set containing unconsolidated requests in route $r$
  Randomly select $Q' \subseteq Q$
  For each $i \in Q'$
    Remove $i$ and $p_{(i)}$ from $r$
    For each $(p, r_m, r_n)$ where $p \in P_{(i)}$ and $(r_m, r_n) \in M'$
      Apply the insertion method to insert $p$ and $i$ into route pair $(r_m, r_n)$
      If successful and cost reduction is observed, break
    End
    If no consolidation happens, update r by inserting back request $i$ and $p_{(i)}$
  End
End
```

Vehicle upgrade-1. As we start with the most economically efficient vehicle, for example f_1, there are trade-offs when a larger vehicle f_2 satisfies $w_{(f_2)} \geq w_{(f_1)}$ and $v_{(f_2)} \geq v_{(f_1)}$ (e.g. a van has a larger capacity than a car but it is slower and less economic efficient in terms of fuel and labor). Consequently, this heuristic aims at merging two routes and assigning a less economically efficient vehicle but with a larger capacity. For simplicity, we define a new operator $Me(r_m, r_n)$ indicating the operation to insert route r_n into route r_m. The newly created route must satisfy time window and capacity constraints as well as the cost reduction condition. Note that, $Me(r_m, r_n) \neq Me(r_n, r_m)$. The approach of merging is presented in Figure 5. It attempts to insert the whole route 2 into route 1 where the insertion position is between vertices b^+ and c^+. After merging, the pickup removal method and de-consolidation method are applied. Pickup removal is to remove some pickup vertices from the route because after merger, some pickup locations can be visited twice since they appear in both routes. As a result we remove the overlapping pickup vertices and require the requests to be picked up at the earliest vertex. In de-consolidation, once there is a successful merge, some consolidated requests are no longer consolidated as its pickup and delivery vertices are in the same route. We alter the flag for consolidation of this request from consolidated to unconsolidated and these requests can be considered for consolidation later. Each time the algorithm applies this neighborhood, only α_4 percent of the route pairs are considered for merging.

The entire step is presented in Algorithm 3.

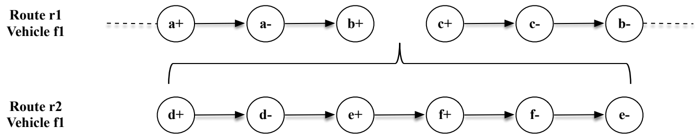

Figure 5. Route merge operator

Algorithm 3: Vehicle Upgrade-1

```
Input:
Set $R$
Initialize randomly select a subset $M'$ from $R\times R$
For each pair $(r_m, r_n)$ in $M'$
  Set $c = c_{(r_m)} + c_{(r_n)}$
  Set $r_{best} = \emptyset$
  For each insertion position $j$ in $r_m$
    Insert $r_n$ at position $j$, denote the combined route as $r$
    Apply Pickup Removal on $r$
    Apply De-consolidation on $r$
    Assign the most efficient feasible vehicle to $r$ (If such a vehicle does not exist, continue.)
    If $(c_{(r)} < c)$ and $r$ is a feasible route
      $c = c_{(r)}$
      $r_{best} \leftarrow r$
    End
  End
  If $r_{best} \neq \emptyset$
    Add $r_{best}$ to $R$ and remove $r_m$ and $r_n$ from $R$
    Break
  End
End
```

Vehicle upgrade-2. Instead of considering merging two routes at one time, this heuristic integrates requests from other routes into one bigger route and replaces the vehicle with a larger one. We first pick one route as a potential vehicle upgrade route and then insert possible requests from other routes into it while keeping the feasibility of the new solution. The condition for a successful insertion is a reduction in total cost. Similarly, we guarantee the feasibility of the routes where requests are removed and also apply the pickup removal and de-consolidation methods on the larger route. Distinct from vehicle upgrade-1, we wait until all beneficial relocations have been made and then compare the cost reduction condition of the new solution to the old solution. The reason is that more beneficial relocations maximize the utilization rate of larger vehicles.We define α_5 as the percentage of requests from each route to be inserted into the larger route.

Vehicle downgrade. In contrast to the two merge heuristics proposed above, splitting a larger route into two smaller routes and assigning a more efficient vehicle is also a potential way in searching for a better solution. Suppose we have route r, and we split it into routes r_1 and r_2. In classic VRP, splitting is to simply break the route at some position in a route and return the vehicle to a depot.

However, in a pickup and delivery situation, simply splitting would lead to "broken requests" (i.e. some requests may not have pickup locations in the new routes). This heuristic has an embedded smoothing process. Suppose there is a split at position h in route r and r_1 is the route split before h while r_2 is after. The smoothing process removes the pickup vertices in r_1 that do not require a visit and adds these pickup vertices to r_2. When adding the pickup vertices to r_2, we keep the order of the vertices unchanged. Also, α_6 is defined as the percentage of routes to be split.

3.3.2 VNS Algorithm

The VNS procedure starts with the initial solution and runs through the different neighborhoods defined before. Before each iteration, the order of neighborhoods is randomly determined. The termination criterion is controlled by the total elapsed running time and the percentage gap between the old and the new solution after each iteration. Algorithm 4 describes the algorithm where N is the set of all 6 neighborhoods.

```
Algorithm 4: Variable Neighborhood Search
Input:
Set $R$, $N$
Initialize denote the current solution as $R_c$, the best solution as $R_b$
Let $R_c \leftarrow R$ and $R_b \leftarrow R$
Loop
  Randomly sort $N$
  For $i = 1$ to $n$
    Apply Local Search $N_i$
    Update solution $R_c$
  End
  Stopping criterion: $\left(c_{(R_b)} - c_{(R_c)}\right)/c_{(R_b)}$ is below or running time exceeds certain thresholds
  Let $c_{(R_b)} = \min\left(c_{(R_c)}, c_{(R_b)}\right)$
  Update solution $R_b$
End
```

3.3 Fleet Constrained Algorithm

To integrate the fleet constraint (i.e. a limited number of fleet vehicles) in the model, we apply a vehicle selection algorithm. Each time when a type of a vehicle is taken into consideration, the algorithm selects the most efficient vehicle with the largest available time window. A vehicle already assigned to some routes is only available 'in-between the routes.' The largest available time window of a vehicle is defined as the window with the largest length the vehicle can serve a route with. Initially, vehicles are ranked by their efficiency. For instance, we can have $F = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$ with the first 3 vehicles from fleet-1, the next 3 of fleet-2 and the last one from fleet-3. The algorithm also maintains list $wl_{(f)}$ to save each a vehicle $f's$ en route time windows. For example $wl_{(f)} = \{(t_1, t_2), (t_3, t_4)\}$ means vehicle f is not available in time windows (t_1, t_2) and (t_3, t_4) since it serves two routes. The vehicle selection process is presented in Algorithm 5. The vehicle selection heuristic has two steps: (1) find the right type of a vehicle; (2) find the vehicle with the largest available time window. It is invoked when considering a vehicle selection or a vehicle change. The output of Algorithm 5 is used as the input in the construction of the initial solution, vehicle upgrade and downgrade heuristics.

```
Algorithm 5: Vehicle Selection
Input:
Sets $F$, $R$, request $i$
Initialize set $A = \emptyset$ to contain time windows
Denote $f^*$ as the optimal vehicle to select and $T^*$ as the largest length available time window for $f^*$to serve a new route
Sort $F$ by $ef_{(f)}$ in the ascending order
For each unprocessed $f$ in $F$
    Let $aw_{(f)}$ be the available window list of $f$ based on $R$
    Iterate $aw_{(f)}$ to find the window $t = (t_e, t_l)$ satisfying condition: $e_{(i)}>t_e$ and $l_{(i)} < t_l$
    If such $t$ exists
        Add $t$ to array $A$
    Else
        Remove $f$ from $F$
    End
End
Let $e$ be the highest efficiency vehicle in the current $F$
Remove entries from $F$ and $A$ whenever entries in $F$ have lower efficiency than $e$
Let $k$ be the position of the largest window length in $A$
Let $f^*$ be the $k\ th$ vehicle in $F$ and let $T^*$ be the $k\ th$ time window in $A$
```

4. Case Study

4.1 Background

The case is based on a U.S. big-box retailer's urban markets. All of the data and parameters were given to us by the retailer and they reflect their current operations and relationships with other stores and clients. We conducted studies on the markets with both medium and high population density. Most of the results are based on a market with medium population density. This market is located in the northern part of Chicagoland and the area is approximately 80 square miles with population density of approximately 6,000 per square mile. The other market is in the New York Manhattan area. We have four different types of pickup locations: a depot, self-owned stores, local merchants and stores requiring shopping. Local merchants are to satisfy customers' requests that can be fulfilled from a local store, for example a nearby corner bakery. Once an order has been placed, local merchants are notified and required to prepare the requests for pickup within a certain window. With regard to stores requiring shopping, the couriers have to shop inside the stores. Due to the differences between pickup locations, two types of service time are used to measure the store pickup time. A fixed service time is applied to the depot, self-owned stores and local merchants. For stores requiring shopping, we formulate a simple model based on the number of requests (roughly 1-10 per store visit) to estimate the shopping time. In terms of the potential pickup location list for each request, we assume if a request is sourced from a chain such as Target, then its potential pickup list includes all locations in the study area. Two types of delivery are considered: delivery to a customer's specified address and customer's self-pickup. For the latter, we assume customers have to visit the depot for self-pickup. As a result, all self-pickup requests are brought back to the depot. With respect to time windows, orders are processed in batch and we define Q_0, Q_1, Q_2, Q_3 as yet undelivered requests from the previous day, morning delivery, noon delivery and evening delivery, respectively. Denote ct_1, ct_2, ct_3 as the cut-off times of orders.

Based on our policy, if a customer places an order by ct_3 (e.g. between 6 pm and 7 pm), then all requests in that order except those requiring customer's self-pickup as well as local store shopping will be pushed to the next day and can be delivered by noon (e.g. between 11 am and 12 pm the next day). So, Q_0 refers to the requests being pushed forward from the previous day. The morning delivery Q_1 means that if the customer places the order before ct_1 (e.g. between 7 am and 8 am) in the morning, then all requests in that order can be delivered to this customer by the end of the day or ready for pickup at the depot before noon. We apply four types of vehicles which are car, van, bicycle and foot. In addition, the two types of problems defined earlier have been studied: (1) the default same-day delivery problem presented in Section 4.3; (2) the fleet constrained problem in Section 4.4. For the default problem, the fleets' parameter settings are presented in Table 2. For the capacity parameters, we set the van's as 100% and present the relative capacity of the other fleets.

Table 2. Fleets' parameter settings

Fleet type	Weight	Volume	Speed (miles/hour)	Wage (dollars/hour)	Fuel (dollars/hour)
Car	16.7%	0.5%	10~20	20~30	0~0.5
Van	100%	100%	10~20	30~40	0~0.5
Bicycle	3.3%	0.05%	5~10	20~30	0
Foot	0.5%	0.02%	0~5	20~30	0

4.2 Data Generation

Data is generated from parameters provided by the retail partner. The data generation process contains three steps: zone creation, distance calculation and request simulation. In zone creation, we divide the study area into squares of $0.25 \text{ mile} \times 0.25 \text{ mile}$. This area corresponds roughly to a block. Each square represents the minimum unit and its center point (measured by longitude and latitude) is considered as the location of one customer. A number of locations are randomly sampled from these squares and orders originate from these samples. We also allow more than one order to originate from the same location. We use PostgreSQL and PostGIS for geographical calculations. In addition, distance computation is sent to MapQuest. Request generation involves several steps. First of all, the number of requests per order is simulated based on the distribution parameters provided by the retailer. Similarly, for each request, we use distribution assumptions to simulate the other characteristics such as the source type of pickup, delivery type (home delivery or self-pickup) and time window. Regarding requests Q_0 from the previous day, we generate requests for two consecutive days. From the orders placed by the cut-off time ct_3 in the first day, we leave out the requests requiring customer's self-pickup as well as local store shopping and push the remaining requests (i.e. Q_0) to the second day. In the second day, from the orders placed by ct_3, we keep the requests requiring customer's self-pickup and local store shopping. These requests are referred to as Q_3. The retail partner operates an e-commerce site and the underlying parameters for orders and requests were determined by analyzing the e-commerce orders. The final dataset has 600-750 requests, 90-120 orders, 60-80 pickup locations and 1 depot. Table 3 describes distribution of requests. Definitions of $\mathrm{P}^0, \mathrm{P}^1, \mathrm{P}^2$ and P^3 are in Table 1 of Appendix. It is clear that Q_1 has the majority of the requests to be delivered and Q_2 is the second largest order placement period.

Table 3. Request distribution

Requests type	P^0	P^1 and P^2	P^3	Total
Q_0	10.0%	3.5%	2.6%	16.1%
Q_1	12.7%	31.7%	14.1%	58.5%
Q_2	7.9%	11.5%	3.8%	23.2%
Q_3	0%	0%	2.2%	2.2%
Total	30.6%	46.7%	22.7%	100%

4.3 Computational Results

The algorithm has been coded in JAVA version 1.6.0 and the experiments are performed on a 2.3 GHz Intel Core i5 processor. The parameters used in this paper are $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \ \alpha_6\} = \{0.2, 1.0, 0.6, 0.4, 0.4, 1.0\}$ and the threshold of VNS running time is $T_{VNS} = 30\ min$. The parameter tuning had two stages: (1) we ran the program with only one neighborhood at a different level of percentages and we found the best setting for this α_i where ties have been observed; (2) all combinations of those potential parameters are implemented and the combination yielding the lowest total cost is chosen. This led to the final choice of α given above.

Based on the provided hourly pay and fuel rate, the solution of the default same-day delivery routing case is between \$1,000 and \$2,000, and Table 4 provides the main results. For confidentiality reason, we cannot provide exact numbers and instead we give ranges. Of significant importance is cost per order. Assuming no profit margin on shipping, this number reflects how much should the retailer charge in terms of the shipping cost (direct charge to orders or bundled within costs of items ordered).

In comparison to other same-day delivery providers, for example, Google Express charges \$4.99 per store for non-members. In our study, the per order per store cost is between \$3.00 and \$4.50. This compares favorably with Google Express (and allowes a solid margin for the provider). This is not unexpected since we leverage the existing infractructure. Assuming the retailer can realize 100 orders per day as we projected, then 6 cars are needed. Suppose the average acquisition cost of a car is around \$15,000-\$25,000, the car's life span is 10 year, and the average yearly maintenance, insurance and other costs together is around \$2000. If we amortize the fixed cost (i.e. annual depreciation, maintenance, insurance, etc.) to a per order per store level, then the incremental cost is around \$0.03-\$0.05 which is very limited.

In the high density area of New York Manhattan it is estimated that cars would be 33% slower and vans 17% slower. With the same number of orders applied in New York Manhattan, we find that the total cost is only slightly lower than the default cost (roughly 1% less). Despite vehicles being slower, higher density implies shorter distances and thus overall cost savings. Since the cost reduction is not substantial, this elicits that same-day delivery service may work better in high population density area if vehicle speeds can be higher than the adjusted setting.

After the VNS improvement, there is a 6.2% reduction in total cost compared to the initial solution. The fleet resource utilization rate is defined as the ratio between the travel time of all vehicles in the solution and their total allowable operating time. In the initial solution, no vehicle upgrade and downgrade operations happen but 10 consolidations are observed. In the VNS part, we observe successful implementations for each neighborhood as follow: 5 intra-route relocation successes; 2 inter-route relocation successes; 22 consolidation successes; 0 vehicle update-1 successes; 0 vehicle update-2 successes; 0 vehicle downgrade successes. The majority of the 6.2% reduction comes from the 22 consolidation successes which yield a 5.5% reduction in the total cost. As there are no improvements made through the vehicle update or downgrade process and fleet resource is unlimited, all vehicles used in the final solution are the most economically efficient vehicles which are cars in this study.

In the study on sensitivity in the next section, vehicle upgrades of both types and downgrades were observed. The number vary from 0 to 3.

Table 4. Computational results summary

Size (# of requests)	600-750
Total cost ($)	1,000-2,000
Cost per order ($)	10-15
Cost structure	Labor 80%-90%, Fuel 10%-20%
En route time structure	Service time 30%-40%, Travel time 60%-70%
Number of cars used	6
Stops per route	20.1
VNS Improvement (%)	6.2
Initial Solution Running Time (seconds)	294
Fleet resource utilization rate (%)	70%-80%
Miles per stop	2-3

4.4 Sensitivity Analyses

The sensitivity analyses are with respect to the fleet structure, cost structure and en route time structure. The default solution is the one presented in Table 4. In this section, we integrate the fleet constraints to the problem. An overview of the resource and rule changes is presented in Table 5.

Table 5. Resource change overview

Problem type	Fleet setting	Change rules	Reference
Fleet constrained	5 cars, 2 vans and at most 4 bicycles	Reduce bicycle hourly pay	See Figure 6
Fleet unconstrained	Unlimited vehicles	Reduce van hourly pay	See Figure 7
Fleet constrained	Limited cars and unlimited bicycles	Reduce number of cars	See Figure 8
Fleet constrained	Limited cars and unlimited vans	Reduce number of cars	See Figure 8 and Figure 9
Service time change	Unlimited cars, vans, bicycles and foot fleets	Change store shopping time and depot handling time	See Figure 10 and Figure 11
Time window change	Unlimited cars, vans, bicycles and foot fleets	Shorten delivery windows	See Figure 12

4.4.1 Fleet Analysis

In the default problem, the car is always the best option due to its highest economic efficiency. Also as the routes in the same-day delivery problem are relatively short, larger vehicles cannot leverage their capacity. For this reason it is valuable to measure: (1) whether reducing one type of vehicle's cost rate makes it enter the final solution; and (2) what is the appropriate reduction ratio. The cost of combustion-engine based vehicles has two components: the labor and fuel costs. Bicycle and foot fleets' costs have only the labor cost. Here we alter the labor cost to change the cost rate. First, by adjusting the hourly pay for bicyclists, we present Figure 6. In this figure, the fleet resource consists of 5 cars, 2 vans and at most 4 bicycles. The reason for these numbers is that 6 cars are the least amount of vehicle resources to yield a solution. We first reduce the number of cars by one and add one more van. Yet, 1 van cannot complete sufficient requests since it has a lower speed than car. In our default problem setting, a van's speed is 20% slower than car's. For this reason 2 vans are appropriate. Based on the analysis shown later in Figure 7, 3 bicycles may replace 1 car so we use 4 bicycles here to prevent an out of resource situation. From Figure 6, we conclude that substantially lowering the hourly bicycle pay barely moves the needle of the total cost. Even if the hourly pay is reduced by 24% (which, for example, is above the Illinois's minimum hourly wage), the reduction in total cost is only 2%. In Figure 6, the x-axis represents the relative reduction of hourly pay for bicyclists. The left y-axis is the difference between the total cost with hourly pay reductions and the default cost-1 (the total cost of the default problem). The right y-axis represents the difference between the initial cost (i.e. without applying the VNS algorithm) with hourly pay reductions and default cost-2 (the initial cost of the default problem). The reduction of hourly pay is not considered beyond 24% since we are not expecting to see the structural change in the initial solution. For example, the initial solution line (dotted line) has a linear trend which means that the fleet combination in the initial solution has not been changed. Once the economic efficiency of bicycles outperforms cars, all routes start to use bicycle in the initial solution.

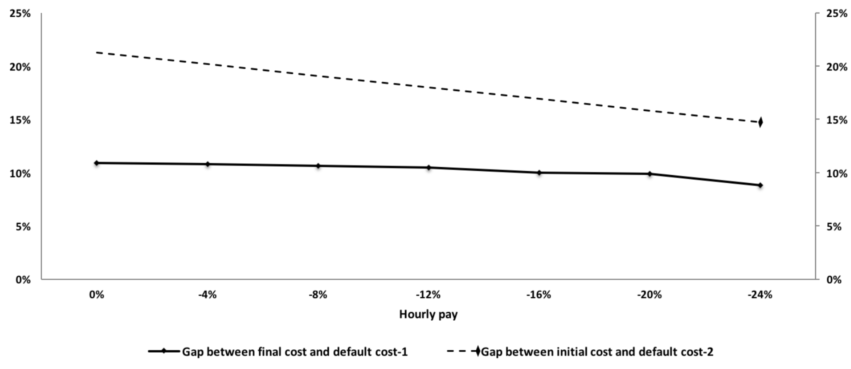

Figure 6. Sensitivity analysis for bicycle's hourly pay

Next we perform an analysis of vans (Figure 7). In this figure, the dotted line represents the trend of total cost in response to the changes of van's hourly pay. The solid line is the number of routes that use vans in the final solution. In the figure, when the hourly pay falls by 51%, there are 2 vans used in the final solution. However, once the hourly pay drops by 54% (which is still above the Illinois's minimum hourly wage), there are 10 vans used in the final solution. The reason for this huge jump is that once the hourly pay drops by too much, van's economic efficiency is higher than car's and is selected as the most favored type when selecting a vehicle. To realize a structural change in fleet utilization, a 51% reduction of van's hourly pay has to be achieved which may not be possible to attract qualified van drivers in practice

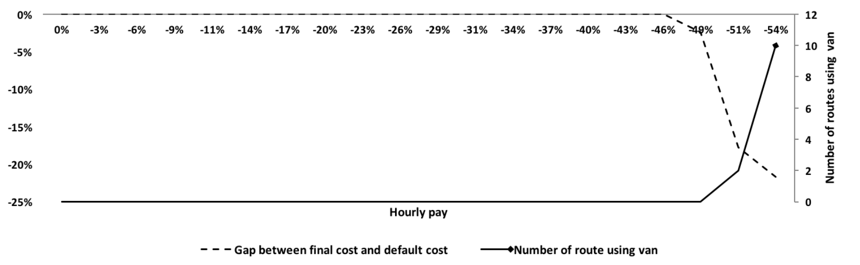

Figure 7. Sensitivity analysis for van's hourly pay

In addition to isolating each type of a vehicle and implementing the analysis independently, we also present studies related to the combination of vehicles. Figure 8 is concentrating on the trade-off between van and bicycle to enter into the final solution. In the default problem, 6 cars are needed to yield a solution. Here the analysis starts by reducing the number of cars one by one and keeping the number of other types of vehicles unlimited to observe how many of them enter into the final solution. In this figure, the black bar stands for the combination of cars and bicycles (i.e. only cars and bicycles are considered) while the white bar is the combination of cars and vans. The dotted line is the default line representing the optimal cost when we have 6 cars. Two key findings can be synthesized from Figure 8. In our cost ratio setting, using vans to replace cars is always better than using bicycles. Second, there is no liner relation between the number of bicycles and cars (e.g. n bicycles replacing 1 car does not imply $2n$ bicycles can replace 2 cars). The first finding follows as the lighter bar is lower than the black bar when we are able to construct solutions for both "car plus bicycle" and "car plus van" scenarios. For the second finding, once we reduce 1 car only 2 bicycles enter the final solution. When 2 cars are dropped, 4 bicycles are used in the final solution. However, this "2 bicycles replace 1 car" condition does not continue when 3 cars are dropped. After further investigation, we found that with only 3 cars the algorithm needs 14 bicycles to find a feasible solution and the total cost for this combination is very high (roughly 105% higher than the default cost).

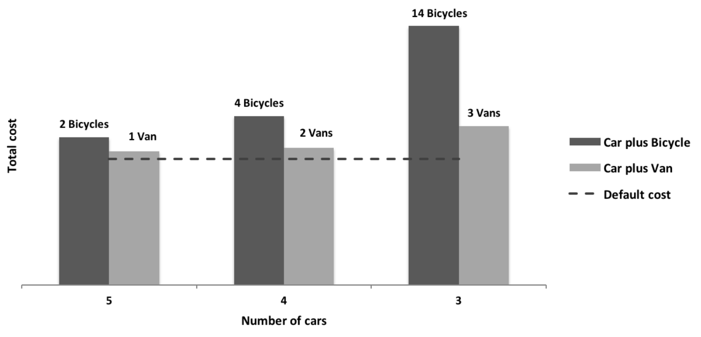

Figure 8. Analysis of vans and bicycles entering the final solution

It is also interesting to investigate when bearing at most $p\%$ increase in total cost, what combination of cars and vans is doable. Figure 9 shows this. The x-axis represents a $p\%$ increase in cost. The first bar of 0% stands for the default solution and it is also the total cost in the default problem. It is interesting to see that there is no combination of cars and vans that can provide a cost increase of 19%. This is due to the fact of using a heuristic and the fact that for each experiment it is started from scratch.

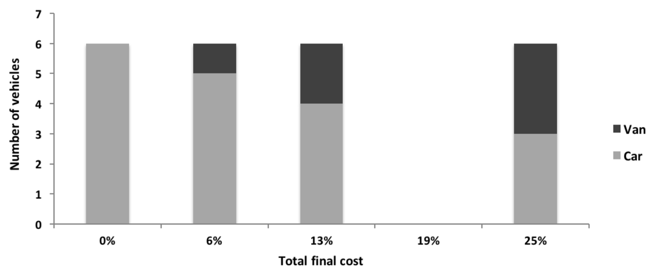

Figure 9. Analysis on the combination of cars and vans

4.4.2 Service and Time Window Analyses

We conduct two distinct time analyses, one centering on the service time and the other are on adjusting the time window constraint. In the solution of the default problem, the service time represents 30%-40% of the total time spent. Service time spent in retailer's own store combined with local merchants is around 20%-30%, in depot handling is around 0%-5% and in local store shopping is around 5%-10%. The labor cost is calculated as the product of labor's hourly pay and working hours. For the time window, providing better service with tighter time windows leads to higher cost. But better service brings potential benefits so it is valuable to investigate cost sensitivity to time windows.

In the analysis of the service time (with the unlimited number of vehicles), we multiply the target, for instance the store shopping time (roughly 5-10 minutes per store), by a scalar β which ranges between [0,1]. The results are displayed in Figures 10 and 11. In Figure 10, we increase the reduction rate by 5% gradually and we find that the reduction of the total cost is only around 2% even when shopping time has been reduced by 25%. In Figure 11, the depot handling time (roughly 10 minutes per route) is reduced by 5% step by step. It shows that when there is a 25% reduction in the depot handling time, total cost can only be reduced by 0.26%. And both analyses imply that the total cost is neither sensitive to store shopping time nor depot handling time. Meanwhile, Figures 10 and 11 both imply that the relations between the two times with the total cost are not linear albeit exhibiting an almost linear trend.

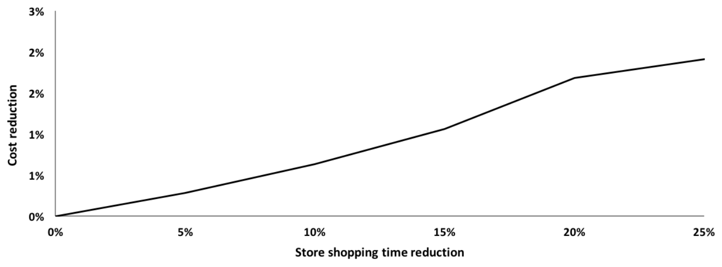

Figure 10. Sensitivity analysis of store shopping time

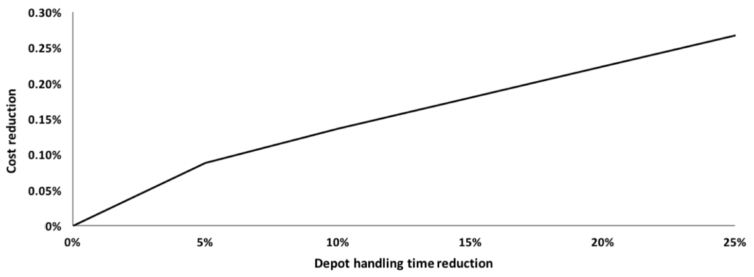

Figure 11. Sensitivity analysis of depot handling time

With regard to the time window, we propose two scenarios. All scenarios have no limits on fleet resources. Time window changes exclude the requests requiring store shopping since most local stores are opened for the majority of a day. In the first scenario, we move the morning home delivery time for requests requiring pickup from either the retailer's own stores or local merchants 8 hour earlier. Similarly, the second scenario is to push forward the noon home delivery time 2 hours earlier for requests requiring pickup at either the retailer's own stores or local merchants. The results are summarized in Figure 12. Intuitively, when the time window between the order placement and delivery is shrunk, cost increases. Two conclusions can be drawn from the analysis: (1) offering better same-day morning delivery (scenario 1) requires 2 more car and increases the total cost by 10%; (2) providing better same-day afternoon delivery (scenario 2) requires 1 more car and increases the total cost by 5%.

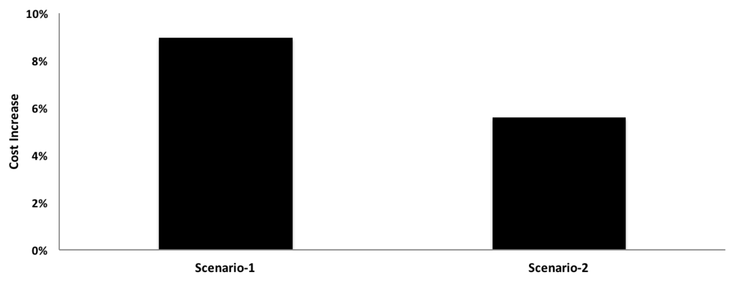

Figure 12. Sensitivity analysis under time window change

Conclusion

The same-day delivery routing problem discussed in this paper adds several new features to the classic VRP. Besides retaining traditional features, this new problem captures mixed fleets, selective pickup and delivery, consolidation and constraints of time windows and capacity. By tackling this problem, a new insertion-based initial solution heuristic considering vehicle change and consolidation has been constructed. Meanwhile, a new VNS algorithm including six adapted neighborhoods has been put forward. The whole algorithm has been tested on a real case based on a U.S. market with a retailer data. Moreover, we performed sensitivity analyses across metrics such as fleet resources, service times and time windows. The results provide insights for the same-day delivery routing business operating by a big-box retailer. The main findings from this case study are summarized as follows.

- Using a depot as a consolidation point helps to reduce the total cost in the fleet unconstrained problem as 5% of total requests can be consolidated and the consolidation heuristic in the VNS algorithm contributes a 5.5% total cost reduction;
- The cost per order is approximately $10-15;
- In a less condensed market, car is better than van or bicycle;
- Moving the morning home delivery time 8 hours earlier for requests requiring pickup from either the retailer’s own stores or local merchants increases the total cost by 10%;
- Pushing forward the noon home delivery time 2 hours earlier for requests requiring pickup at the retailer’s own stores or local merchants increases the total cost by 5%;
- Under the same order distribution but with decrease speed of select vehicles, total cost in a high population density area is slightly lower than in a medium population density area.

Future studies include improving the current VNS algorithm to yield better solutions and adapting the algorithm to a real-time order arrival setting. Meanwhile, currently we are using a single point for consolidation, however it might be more beneficial to include more points.

References

Baldacci, R., Battarra, M., Vigo, D. 2009 Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs. Networks. 54: 178-189.

Bräysy, O., W. Dullaert, G. Hasle, D. Mester, M. Gendreau. 2008. An effective multirestart deterministic annealing meta-heuristic for the fleet size and mix vehicle routing problem with time windows. Transportation Science. 42: 371-386.

Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G. 2007. Static pickup and delivery problems: A classification scheme and survey. TOP. 15: 1-31.

Choi, E., Tcha, D.W. 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. Computers and Operations Research. 34: 2080-2095.

Emeç, U., Çatay, B., Bozkaya, B. 2016. An adaptive large neighborhood search for an E-grocery delivery routing problem. Computers and Operations Research. 69: 109-125.

Gendreau, M., Laporte, G., Musaraganyi, C., Taillard, É.D. 1999. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. Computers and Operations Research. 26: 1153-1173.

Gribkovskaia, I., Laporte, G., Shyshou A. 2008. The single vehicle routing problem with deliveries and selective pickups. Computers and Operations Research. 35: 2908-2924.

Golden, B.L., Assad, A.A., Levy, L., Gheysens, F. 1984. The fleet size and mix vehicle routing problem. Computers and Operations Research. 11:49-66

Hanse, P., Mladenović, N. 2001. Variable neighborhood search: Principles and applications. European Journal of Operational Research. 130: 449-467.

Jabali, O., Gendreau, M., Laporte, G. 2012. A continuous approximation model for the fleet composition problem. Transportation Research Part B: Methodological. 46: 1591-1606.

Koç, Ç., Bektaş, T., Jabali, O., Laporte, G. 2016. Thirty years of heterogeneous vehicle routing. European Journal of Operational Research. 249: 1-21.

Lee, Y.H., Kim, J.I., Kang, K.H., Kim, K.H. 2008. A heuristic for vehicle fleet mix problem using tabu search and set partitioning. Operational Research Society. 59: 833-841.

Masson, R., Lehuédé, F., Péton, O. 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transportation Science. 47: 344-355.

Mitrović-Minić, S., Laporte, G. 2006. The pickup and delivery problem with time windows and transshipment. INFOR. 44: 217-228.

Mladenović, N., Hansen, P. 1997. Variable neighborhood search. Computers and Operations Research. 24: 1097-1100.

Muses, C., Pickl, S., Bein, W., Chin, F.Y.L., Hsu, D.F., Palis, M.L. 2005. Transshipment and time windows in vehicle routing. Proceedings of the 8th International Symposium on Parallel Architectures. Algorithms and Networks, IEEE Computer Society, Washington, D.C. 113-119.

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

Osman, I., Salhi, S. 1996. Local search strategies for the vehicle fleet mix problem. Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., Smith, G.D. eds. Modern Heuristic Search Methods. Wiley, New York. 131-153.

Pessoa, A., Uchoa, E., Poggi de Aragão, M. 2009. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. Networks. 54: 167-177.

Petersen, H., Ropke, S. 2011. The pickup and delivery problem with cross-docking opportunity. Böse, J., Hu, H., Jahn, C., Shi, X., Stahlbock, R., VoB, S. eds. Computational Logistics Lecture Notes in Computer Science. 6971: 101-133.

Qu, Y., Bard, J.F. 2012. A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. Computers and Operations Research. 39: 2439-2456.

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

Salhi, S., Rand, G.K. 1993. Incorporating vehicle routing into the vehicle fleet composition problem. European Journal of Operational Research. 66: 313-330.

Shang, J.S., Cuff, C.K. 1996. Multicriteria pickup and delivery problem with transfer opportunity. Computers and Industrial Engineering. 30: 631-645.

Taillard, É.D. 1999. A heuristic column generation method for the heterogeneous fleet vehicle routing problem. RAIRO (Recherche Opérarionnelle/ Operations Research). 33: 1-14.

Thangiah, S.R., Fergany, A., Awam, S. 2007. Real-time split-delivery pickup and delivery time window problems with transfers. Central European Journal of Operations Research. 15: 329-349.

Toth, P., Vigo, D., eds. 2002. The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia.

Wassan, N.A., Osman, I.H. 2002. Tabu search variants for the mix fleet vehicle routing problem. Journal of the Operational Research Society. 53: 768-782.

Yanik, S., Bozkaya, B., deKervenoael, R. 2014. A new VRPPD model and a hybrid heuristic solution approach for e-tailing. European Journal of Operational Research. 236:879-890.

Appendix

Table 1. Nomenclature

Name	Description
Set Q	Set Q contains all request i with hard time window $[e_{(i)}, l_{(i)}]$
Set F	Set F contains all vehicles f_k sorted by economic efficiency $ef_{(f_k)}$
Set R	Set R contains all routes
Set N	Set N contains n neighborhoods
Set α	Set α contains all parameters corresponding to all neighborhoods
$[s_{(r)}, rb_{(r)}]$	Route $r's$ start time and return time
$P_{(i)}$	Set containing allowable pickup locations for request i
$p_{(i)}$	Pickup location for request i
$c_{(r)}$	Cost of route r
$c_{(R)}$	Total cost for all routes in R
$wl_{(f)}$	A set of $f's$ en route time (i.e. time elapsed between the instant the vehicle leaves the depot to the time it returns) windows based on the current solution
$aw_{(f)}$	A set of $f's$ not en route time windows based on the current solution
$v_{(f)}$	Volume capacity of fleet f
$w_{(f)}$	Weight capacity of fleet f
P^0	A set of requests to be picked up at the depot
P^1	A set of requests to be picked up at stores owned by retailer
P^2	A set of requests to be picked up at local merchant's own stores
P^3	A set of requests to be shopped at other stores