CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Integrating Dock-Door Assignment and Vehicle Routing in Cross-Docking

Furkan Enderer
Claudio Contardo
Ivan Contreras

April 2017

CIRRELT-2017-21

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Integrating Dock-Door Assignment and Vehicle Routing in Cross-Docking

Furkan Enderer[1,2], Claudio Contardo[1,3,*], Ivan Contreras[1,4]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

[4] Department of Mechanical and Industrial Engineering, Concordia University, 1515 St-Catherine Street W., EV4 139, Montréal, Canada H3G 1M8

Abstract. This paper presents an integrated cross-dock door assignment and vehicle routing problem arising in the operation of cross-dock terminals. It consists of assigning origins to inbound doors, transferring commodities between doors, and routing vehicles from outbound doors to destinations. The objective is to jointly minimize the total material handling and transportation costs. Two formulations of the problem are presented and computationally compared. In addition, we develop a column generation algorithm based on the most promising formulation and a heuristic to obtain lower and upper bounds on the optimal solution of the problem, respectively. Numerical results on a set of benchmark instances with up to 20 origins and 50 destinations confirm the efficiency of the proposed solution algorithms.

Keywords: Cross-docking, vehicle routing, dock-door assignment, column generation.

Acknowledgment. This research has been partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds de recherche du Québec - Nature et technologies (FRQNT). These supports are gratefully acknowledged. The authors also thank the two anonymous reviewers for their valuable comments.

---

* Corresponding author: Claudio.Contardo@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2017

1 Introduction

One of the most appealing supply chain strategies that has recently gained global recognition is cross-docking. Many companies throughout different industries from manufacturing to retailing have put their attention on improving the efficiency of their logistics operations by implementing such consolidation strategy. Its key feature is the use of high velocity distribution centres, referred to as *cross-dock terminals*, in which goods arriving from several origins are unloaded from inbound trucks, consolidated and handled according to their destinations, and then loaded into outbound trucks leaving for their destinations with little or no storage in between. Unlike a traditional warehousing approach, cross-docks do not serve storage purposes. An efficient cross-docking strategy seeks to reduce or eliminate storage and material handling costs by keeping little or no storage in the cross-dock and by achieving a perfect synchronization for consolidation. Distribution companies and suppliers benefit from these facilities in many ways, such as reducing storage space, while having immediate responses to the supply chain fluctuations. These facilities improve the efficiency of supply chains and the distribution management of goods by eliminating or minimizing many non-value attached operations such as product movements and storage. Nowadays, cross-docks are implemented and managed efficiently from small-scale companies to large suppliers and logistic providers.

The first cross-docking terminals date back to 1930's and were introduced by the US trucking industry and then around 1950's by the US army (Arnaout et al., 2010). When Walmart adopted cross-docking in the 1980s, it became a huge success in the retail industry and showed an example of how to apply this strategy efficiently. Its cross-docking practices are known to be one of the most efficient implementations of this logistics strategy in supply chain management (Chandran, 2003).

Cross-docking has now become a common practice in the manufacturing and retailing industries such as package delivery providers, air transportation, automobile industry, and less-than-truck-load (LTL) providers. In the case of package delivery, companies receive incoming shipments at cross-docks, sort the packages and ship them out as soon as possible. Given the high standards of service time guarantees, packages move quickly through the network, hardly spending more than a couple of hours inside a cross-dock. Federal Express, the United Postal Services, and the US Postal Service are prototypical examples of the cutting edge in cross-docking (Apte and Viswanathan, 2000).

Given the relevance of cross-docking in modern supply chains, several classes of decision problems have been studied in the literature. Belle et al. (2012) present a comprehensive review and classification of problems arising in cross-docking. Some strategical problems deal with determining the optimal layout and the location of cross-docks, whereas other tactical decision problems consider optimizing the flow of products through a network of cross-docks. Operational decision problems are abundant and arise in different situations such as truck scheduling, dock-door assignment, vehicle routing, workforce assignment, and the location of goods in the temporary storage area. Most of these problems have been studied independently of each other, even though their decisions are naturally interrelated. Buijs et al. (2014) provide a recent survey of cross-docking networks that highlights the need for integration and synchronization of such decision problems. For additional reviews on various cross-docking related

topics, we refer to Boysen and Fliedner (2010) and Agustina et al. (2010).

In this paper we study a new problem in cross-docking denoted as the *Dock-Door Assignment and Vehicle Routing Problem* (DAVRP). The DAVRP integrates within the decision-making process two important operational decisions: the assignment of trucks to dock-doors and the design of vehicle routes. Incoming trucks arriving at the cross-dock from several origins and carrying a family of products are assigned to strip doors. Products are then unloaded, sorted and consolidated according to their destinations. Using material handling equipment, these products are transferred from strip to stack doors to be loaded onto outgoing trucks. Material handling keeps on operating until all the products are transferred to stack doors and loaded onto outgoing trucks that serve their corresponding destinations. Once the outgoing trucks are loaded with the required products, they leave the stack doors, visit a subset of destinations and come back to the cross-dock by the end of the operation. The DAVRP focuses on the following decisions: *i)* the assignment of inbound trucks to strip doors, *ii)* the transferring of products from strip to stack doors, *iii)* the assignment of destinations to stack doors, and *iv)* the design of vehicle routes for outbound trucks. The objective is to jointly optimize the material handling cost at the cross-dock to transfer products from strip to stack doors and the transportation cost of outbound trucks to route the products to their destinations. To the best of our knowledge, cross-docking models integrating dock-door assignment and vehicle routing decisions have not yet been address in the literature.

Potential applications of the DAVRP arise in the retail industry, where companies use cross-dock terminals to consolidate and redistribute products coming from a set of suppliers and destined to several retail stores. Retail companies may use their own logistics infrastructure (cross-docks, fleet of trucks and containers, etc) or a third-party logistic provider to do so. A concrete example of an application of the DAVRP appears in the case when suppliers are responsible for sending the products to the cross-dock. In particular, each supplier sends full truckloads, containing a mix of products destined to several retail stores, from their manufacturing facilities and/or distribution centers directly to the retailer's cross-dock. Transportation costs associated with the routing of products before arrival to the cross-dock are either incurred by the suppliers or considered as part of the fixed ordering costs. At the cross-dock, each inbound truck is then unloaded, cross-docked, and outbound trucks are loaded. Retailers then transport these products to the appropriate stores using their own fleet of trucks. Given that the demand for each store is usually not enough for full truckloads, products destined to several stores are consolidated into a single outbound truck. Retail companies need thus to manage and optimize their internal operations (consolidation, sorting, and handling) as well as the outgoing shipments.

Product consolidation at inbound trucks affects the consolidation of outbound trucks in terms of material handling costs. Moreover, the consolidation of outbound trucks affects the best possible sequence in which the customers are visited in terms of transportation costs. As highlighted by Buijs et al. (2014), effective cross-docking is achieved through the synchronization of local and network-wide operations, and the outputs of these operations affect one another. On the operational level, synchronization corresponds to interdependencies between cross-dock scheduling and network scheduling. When the local and network-wide operations are handled disjointly, i.e., when dock-door assign-

ment decisions are taken independently of the design of the outgoing trucks, the operations may not only lead to an increase of the overall costs but also to inappropriate, or even infeasible schedules. These decisions should thus be jointly optimized. Naskaris et al. (2014) presents a case study in which a cross-dock strategy is evaluated to optimize the supply of local produce to grocery stores in North Carolina, USA. The authors show that both internal handling costs at the cross-dock and outbound transportation costs play an important role on the overall cost.

The main contributions of this paper are as follows. We introduce the DAVRP, a new cross-docking model that integrates two interrelated operational decisions, namely the material handling and vehicle routing decisions. We present two mixed integer programming (MIP) formulations for the DAVRP, a multi-commodity flow based formulation and a configuration based formulation. We develop a column generation algorithm based on the most promising formulation to efficiently obtain lower bounds. Moreover, we present a heuristic algorithm that exploits the information generated by the column generation and takes advantage of the special structure of the problem to efficiently explore the solution space by solving a series of *generalized assignment problems* (GAP). We analytically compare the two presented formulations as well as the solutions' quality obtained with the proposed heuristic. Finally, we analyze the value of using an integrated approach for the dock-door assignment and vehicle routing decisions as compared to two sequential approaches in which these decisions are independently made.

The remainder of this paper is organized as follows. Section 2 reviews the most relevant literature related to the DAVRP. Section 3 presents the formal definition and modeling assumptions, whereas Section 4 introduces two mathematical programming formulations for the DAVRP. In Section 5, we describe the proposed column generation algorithm. In Section 6, we present a heuristic for the DAVRP. The results of computational experiments performed on a set of randomly generated instances as well as some managerial insights associated with the integrated solution approach are given in Section 7. Conclusions follow in Section 8.

2 Literature Review

The DAVRP is closely related to two classes of cross-docking models: the *Cross-Dock Door Assignment Problems* (CDAP) and the *Vehicle Routing Problems in Cross-Docking* (VRPCD). The CDAP deals with a single cross-dock in which a set of origins must be assigned to inbound doors and a set of destination points must be assigned to outbound doors such that the material handling cost inside the cross-dock is minimized. The cost associated with dock-door assignment problems is that of transporting the goods from inbound doors to outbound doors and is often represented as the traveling distance between doors. The CDAP was first addressed by Peck (1983). The authors consider capacities on inbound and outbound doors. The objective is to minimize the total time spent by transferring products from inbound to outbound doors. In Tsui and Chang (1990), the authors study a CDAP model where each inbound door is assigned to only one origin and each outbound door to only one destination, respectively. Tsui and Chang (1992) propose a branch-and-bound method to solve the prob-

lem studied by Tsui and Chang (1990). Zhu et al. (2009) extend the existing CDAP models to consider a more realistic case where the number of origins and destinations is much larger than the number of inbound and outbound doors. Moreover, they introduce capacity constraints on each door. The problem is formulated as a nonlinear integer program and solved using a branch-and-bound algorithm. Guignard et al. (2012) introduce heuristics to solve the model introduced in Zhu et al. (2009). Nassief et al. (2016) propose an MIP formulation for the CDAP that is solved by means of Lagrangean relaxation to derive lower and upper bounds. Nassief et al. (2017) study several MIP formulations for the CDAP which are theoretically and computationally compared. For additional works on the CDAP we refer to Oh et al. (2006) and Bozer and Carlo (2008). We also refer to Luo and Noble (2012), Liao et al. (2013), Kuo (2013), Choy et al. (2012), and Nassief et al. (2017) for other CDAPs that incorporate additional features such as limited capacity on storage and staging areas and scheduling decisions.

The VRPCD deals with the pick-up of products from a set of origins and the delivery of such products to destinations through a single cross-docking facility. The objective is to minimize the total cost of routing vehicles from the cross-dock to origin points and of routing vehicles from the cross-dock to the corresponding destinations. The cost is often referred to as the total traveling cost. The VRPCD was initially introduced by Young et al. (2006), were the authors assume that inbound vehicles arrive at the cross-dock simultaneously after the pick-up and these vehicles are then routed from the cross-dock to destination points for delivery. The vehicles are assumed to have limited capacities. However, the inbound and outbound doors are assumed to have unlimited capacity to process all the vehicles. The authors present a MIP formulation and develop a tabu search algorithm to obtain upper bounds on the optimal solution value of VRPCD. Another tabu search algorithm is introduced by Liao et al. (2010) and compared against the one of Young et al. (2006). Wen et al. (2009) consider the same problem without the synchronization constraint. Instead, consolidation decisions are used to determine dependency among vehicles. Moreover, they integrate time windows for each pickup and delivery. A MIP formulation and a tabu search algorithm is presented. For similar works introducing heuristic algorithms we refer to Tarantilis (2013), Dondo and Cerdá (2013) and Morais et al. (2014). Santos et al. (2011a) study a model that integrates vehicle dependencies by penalizing the unloaded requests at the cross-dock. The authors present a set partitioning formulation and a branch-and-price algorithm to solve it. Santos et al. (2011b) introduce an alternate formulation for the same problem by representing pick-up and delivery routes with a single variable. The resulting problem is solved using branch-and-price. The new algorithm provides better bounds and is capable to prove optimality for a larger number of instances. Santos et al. (2013) extend the works of Santos et al. (2011b) and Santos et al. (2011a) by allowing the pickup trucks to serve the customers without stopping at the cross-dock. The authors implement a branch-and-price algorithm to solve the problem. Agustina et al. (2014) developed an integrated model for a food retailer company that is responsible of handling the internal operations (inventory) and outgoing shipments.

We would like to conclude this section by highlighting that the existing CDAP literature focuses on the material handling costs, whereas VRPCD focus only on the transportation costs. Despite the fact that material handling and

transportation decisions are interrelated, these two costs have been optimized disjointly in the literature. The proposed DAVRP integrates these two classes of decision problems to jointly optimize the material handling and transportation costs. As we will show in Section 7, integrating these two decisions may provide substantial gains in terms of cost reduction, as compared to a sequential — unintegrated— optimization process.

3 The Dock-Door Assignment and Vehicle Routing Problem

Let $G = (V, A)$ be a directed graph where V denotes the set of vertices and A denotes the set of arcs. The node set V is composed of four pairwise disjoint subsets I, J, M and N. The sets I and J represent, respectively, the inbound and the outbound doors, whereas the sets M and N represent, respectively, the origin points (suppliers) and the destination points (customers). The arc set A is also composed of four pairwise disjoint subsets A_1, A_2, A_3 and A_4. Set $A_1 = \{(m, i) : m \in M, i \in I\}$ represents the arcs connecting each possible origin point with each inbound door. Set $A_2 = \{(i, j) : i \in I, j \in J\}$ represents the arcs connecting each inbound door to each outbound door. Set $A_3 = \{(j, a), (a, j) : j \in J, a \in N\}$ represents the arcs connecting outbound doors with destination vertices. Finally, set $A_4 = \{(a, b), (b, a) : a, b \in N, a < b\}$ represents the arcs connecting each pair of destinations. Let K be the set of commodities. For each $k \in K$, let $o(k) \in M$ and $d(k) \in N$ be the origin and the destination nodes of commodity k, respectively. Also, let q_k be the amount (load) of the commodity. For every $m \in M$, let $O_m = \sum_{k \in K: o(k)=m} q_k$ be the total amount of goods originating at m.

For each arc $(i, j) \in A_2$, let C_{ij} be the unit handling cost for transferring one unit of commodity from inbound door i to outbound door j. For every arc $(a, b) \in A_3 \cup A_4$ let T_{ab} be the traveling cost associated to the use of arc (a, b). Furthermore, let us denote by Q the homogenous fleet size. For each $i \in I$ and $j \in J$, let Q_i and Q_j be the capacity of inbound and outbound doors, respectively. Outbound door capacities are assumed to be always greater than or equal to the vehicle capacity.

We assume that suppliers are responsible for sending the products to the cross-dock and thus, we disregard the traveling costs from origin points to inbound doors. We also assume that the traveling costs from the cross-dock terminal to the destination points are independent from the outbound door, this is $T_{jb} = T_{j'b}$ for each $j, j' \in J$ and $b \in N$.

The DAVRP consists of assigning origin nodes to inbound doors, of transferring commodities from inbound to outbound doors, and of routing such goods to destination points. The objective is to minimize the total material handling and transportation costs, while respecting inbound and outbound door capacities and vehicle capacities.

Figure 1 depicts a graphical representation of a possible solution to the DAVRP for an instance with four suppliers, eight commodities, two inbound and outbound doors and six customers. In this figure, commodity 4 follows path $M_2 - I_1 - J_2 - N_4 - N_5 - N_6$ and commodity 8 follows the path $M_4 - I_2 - J_2 - N_4 - N_5 - N_6$. These two commodities originate at different origin points but

have a common destination point, and they are carried by the same vehicle to their destination. Each commodity has a unique origin and a unique destination

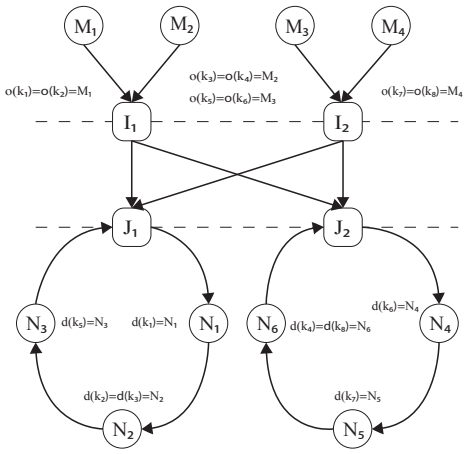

Figure 1: Graphical Representation of a DAVRP instance.

point as well as a corresponding quantity. However, an origin or destination point may be associated with more than one commodity. From an application view-point, this corresponds to suppliers providing different commodities to a subset of customers and customers demanding products from more than one supplier. A destination node having more than one commodity leads to two different approaches in vehicle routing. For example, assume that two different commodities sharing the same destination node would end up being in different outbound doors. In such a case, two vehicles would be needed in two different outbound doors to deliver these commodities to the same destination node. That is, two vehicles would provide service to a single destination node (i.e., split deliveries are allowed). If not preferred, one would need to enforce that two commodities associated to the same destination node must be available at the same outbound door, so as to be transported to their destination node by the same vehicle. The latter case is much more restrictive than the former, and so we assume that split deliveries on the destination nodes are allowed. To model this feature we perform a pre-processing on the destination nodes. Each destination node is duplicated in such a way that there is a single destination node for every single commodity in the network. From now on we assume that each destination node $b \in N$ is associated to a single commodity, and we denote such commodity as $k(b)$.

4 MIP Formulations for the DAVRP

In this section we introduce two MIP formulations for the DAVRP. We first introduce a multi-commodity flow based formulation of the problem that uses continuous flow variables to track the routing of commodities through the network. We then introduce a configuration based formulation that exploits the sub-structures that can be found in the problem to define inbound assignment

patterns, routing configurations patterns between doors, and routing patterns using two huge sets of binary variables.

4.1 Multicommodity Flow Based Formulation

The multi-commodity flow based formulation for the DAVRP can be seen as an adaptation to the formulation introduced by Garvin et al. (1957) for vehicle routing problems in the context of oil delivery. The DAVRP can be defined using the following set of decision variables. Let $X_{mi}, (m,i) \in A_1$ be binary variables denoting the assignment of origin points to inbound doors such that $X_{mi} = 1$ if and only if origin m is assigned to inbound door i. Let $Y_{ijk}, (i,j) \in A_2, k \in K$ be binary variables denoting the assignment of commodities from inbound doors to outbound doors such that $Y_{ijk} = 1$ if and only if commodity k is transferred from inbound i to outbound j. Let $Z_{abj}, (a,b) \in A_3 \cup A_4, j \in N$ be binary variables denoting the routing decisions associated with outbound doors. Let $Z_{abj} = 1$ if and only if a vehicle departing from outbound door j traverses the edge connecting nodes a and b. Let $R_{abl}, (a,b) \in A_3 \cup A_4, l \in N$ be additional flow variables that specify the amount of demand destined to customer $l \in N$ that is transported using arc (a,b). For notational convenience, we assume that each outbound door $j \in J$ is associated with a dummy commodity with zero demand, this is $q_{k(j)} = 0$. The DAVRP can be formulated as follows:

$$\text{(F1) minimize} \quad \sum_{k \in K} \sum_{(i,j) \in A_2} C_{ij} q_k Y_{ijk} + \sum_{j \in J} \sum_{(a,b) \in A_3 \cup A_4} T_{ab} Z_{abj}$$

$$\text{subject to} \quad \sum_{i \in I} X_{mi} = 1 \qquad m \in M \tag{1}$$

$$X_{o(k)i} = \sum_{j \in J} Y_{ijk} \qquad i \in I, \forall k \in K \tag{2}$$

$$\sum_{j \in J} \sum_{k \in K} q_k Y_{ijk} \leq Q_i \qquad i \in I \tag{3}$$

$$\sum_{a \in (N \cup J) \setminus \{d(k)\}} Z_{ad(k)j} = \sum_{i \in I} Y_{ijk} \qquad j \in J, k \in K \tag{4}$$

$$\sum_{j \in J} \sum_{b \in (N \cup J) \setminus \{a\}} Z_{abj} = 1 \qquad a \in N \tag{5}$$

$$\sum_{a \in (N \cup J) \setminus \{n\}} Z_{anj} - \sum_{a \in (N \cup J) \setminus \{n\}} Z_{naj} = 0 \qquad n \in N, j \in J \tag{6}$$

$$\sum_{k\in K}\sum_{a\in(N\cup J)\setminus\{d(k)\}} q_k Z_{ad(k)j} \le Q_j \qquad j\in J \tag{7}$$

$$\sum_{a\in(N\cup J)\setminus\{l\}} (R_{all} - R_{lal}) = q_{k(l)} \qquad l\in N \tag{8}$$

$$\sum_{a\in(N\cup J)\setminus\{b\}} (R_{abl} - R_{bal}) = 0 \qquad b,l\in N, b\neq l \tag{9}$$

$$\sum_{b\in J}\sum_{a\in(N\cup J)\setminus\{b\}} (R_{abl} - R_{bal}) = -q_{k(l)} \qquad l\in N \tag{10}$$

$$\sum_{b\in(N\cup J)\setminus\{a\}} \Big(\sum_{l\in N} R_{abl} - (Q - q_{k(a)})\sum_{j\in J} Z_{abj}\Big) \le 0 \qquad a\in N\cup J \tag{11}$$

$$R_{abl} \le D_l \sum_{j\in J} Z_{abj} \qquad (a,b)\in A_3\cup A_4,\ l\in N \tag{12}$$

$$Y_{ijk}\in\{0,1\} \qquad i\in I, j\in J,\ k\in K \tag{13}$$

$$X_{mi}\in\{0,1\} \qquad m\in M, i\in I \tag{14}$$

$$Z_{abj}\in\{0,1\} \qquad (a,b)\in A_3\cup A_4,\ j\in J \tag{15}$$

$$R_{abl}\ge 0 \qquad (a,b)\in A_3\cup A_4,\ l\in N \tag{16}$$

The objective function minimizes the cost occurring due to the flow of goods from inbound doors to outbound doors and the transportation cost of the goods from outbound doors to customers. Constraints (1) are the degree constraints imposing that every supplier vertex m must be assigned to an inbound door i. Constraints (2) denote that if an origin vertex m is assigned to an inbound door i, then all the commodities coming from that origin vertex must be handled through inbound door i and must be assigned to an outbound door. Constraints (3) are the capacity constraints for inbound doors. Constraints (4) are the linking constraints ensuring that exactly one of the vehicles leaving the inbound door j travels to the destination of a commodity k, if that commodity is assigned to the outbound door j. Constraints (5) force every customer $n \in N$ to be served exactly once, flow conservation constraints (6) denote that if a vehicle is entering a vertex, it must also leave the vertex, and constraints (7) make sure that outbound door capacities are respected. Constraints (8)-(12) make sure that there will be no sub-tours and that the vehicle capacities will be respected. Finally constraints (13)-(15) impose the integrality conditions and (16) impose the non-negativity of the flow variables.

4.2 A Configuration Based Formulation

For each door $i \in I$, we denote $\Omega_i^P = \left\{M^{'} \subseteq M : \sum_{m\in M'} O_m \le Q_i\right\}$ the set of subsets of origins which are assigned to strip door i which satisfy the capacity constraints. We define the binary parameter a_m^p equal to 1 if and only if supplier m is assigned to inbound door i in pattern p. In addition, for a given door i, configuration $p \in \Omega_i^P$, and commodity $k \in K$ such that $o(k) \in p$, we define the

binary parameter h^p_{jk} equal to one if and only if commodity k is routed through outbound door j in configuration p. For each $i \in I$ and $p \in \Omega^P_i$, we define binary variables θ^p equal to 1 if and only if configuration p is selected for door i. Let $C^p_P = \sum_{k \in K: o(k) \in p} \sum_{j \in J} h^p_{jk} C_{ij} q_k$ the handling cost of configuration $p \in \Omega^P_i$.

Figure 2 depicts two different configuration patterns for an inbound door I_1. In the figure, the two origin nodes M_1 and M_2 are assigned to inbound door I_1 for both patterns. The commodities k_1, k_2, k_3 and k_4 are then assigned using different outbound doors to be shipped later by routes.

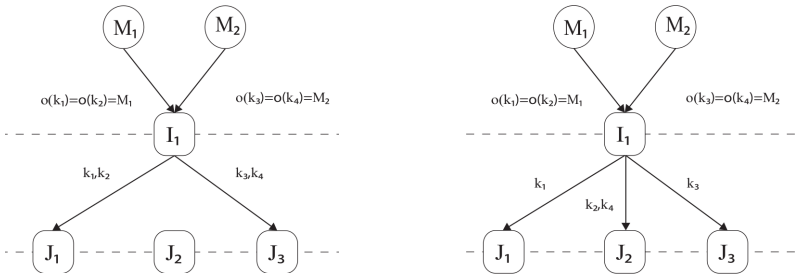

Figure 2: Two different assignment patterns

For each outbound door $j \in J$, let Ω^R_j be the set containing all feasible routes departing from and arriving to outbound door j. A feasible route $r \in \Omega^R_j$ defines a structure such that a vehicle leaves the cross-dock from door j, visits a subset of customers whose accumulated demand does not exceed the vehicle capacity, and then returns to the same outbound door at the end of the operation. The cost of the route r is equal to the sum of the routing costs along the edges traversed by the vehicle, and is denoted by C^r_R. We define the binary parameter b^r_n equal to 1 if and only if customer n is visited by route r and the parameter g^r_e equal to 1 if and only if route r uses edge e along its route. For each $j \in J$ and $r \in \Omega^R_j$, we define binary decision variables λ^r equal to 1 if and only if routing configuration r used at outbound door j. Using decision variables λ^r and θ^p, the DAVRP can be stated as follows:

$$\text{(F2) Minimize} \qquad \sum_{i \in I} \sum_{p \in \Omega^P_i} C^p_P \theta^p + \sum_{j \in J} \sum_{r \in \Omega^R_j} C^r_R \lambda^r$$

$$\text{subject to} \qquad \sum_{i \in I} \sum_{p \in \Omega^P_i} a^p_m \theta^p = 1 \qquad m \in M \tag{17}$$

$$\sum_{p \in \Omega^P_i} \theta^p \leq 1 \qquad i \in I \tag{18}$$

$$\sum_{i\in I}\sum_{p\in\Omega_i^P} h_{jk}^p\theta^p - \sum_{r\in\Omega_j^R} b_{d(k)}^r\lambda^r = 0 \qquad j\in J, k\in K \tag{19}$$

$$\sum_{j\in J}\sum_{r\in\Omega_j^R} b_n^r\lambda^r = 1 \qquad n\in N \tag{20}$$

$$\sum_{r\in\Omega_j^R}\left(\sum_{n\in N} q_{k(n)}b_n^r\right)\lambda^r \le Q_j \qquad j\in J \tag{21}$$

$$\lambda^r\in\{0,1\} \qquad j\in J, r\in\Omega_j^R \tag{22}$$

$$\theta^p\in\{0,1\} \qquad i\in I, p\in\Omega_i^P. \tag{23}$$

The objective function aims at minimizing the total cost. Constraints (17) ensure that every origin point is assigned to an inbound door, while constraints (18) make sure that there is at most one assignment pattern associated with each inbound door. Constraints (19) ensure that if a commodity k is assigned to outbound door j, then there must be a route departing from j and visiting the corresponding destination $d(k)$ of the commodity. Constraints (20) denote that every customer must be visited exactly once and constraints (21) ensure that outbound door capacities are respected.

5 A Column Generation Algorithm

Column generation (CG) is an optimization technique suited to solve linear problems with a huge number of variables. The main idea behind CG is to divide the original linear program, denoted as the *master problem* (MP), into two inter-related subproblems: a *restricted master problem* (RMP) and a *pricing problem* (PP). Given the large number of columns (or variables) in the MP, the RMP contains a small subset of them at every iteration, which is dynamically enlarged. At any given iteration, the RMP is solved by means of, for instance, the simplex method. As solving this RMP is equivalent to solving the MP with many variables fixed to zero (non-basic), the PP is executed to find a non-basic variable of negative reduced cost to enter the basis, or to prove that no such variable exists. The process is repeated as long as the PP succeeds in finding new variables of negative reduced cost. In this section we show how a CG algorithm can be used to solve the LP relaxation of *F2*.

5.1 The Restricted Master Problem

We denote as Ω_{it}^P the subset of feasible configuration patterns for inbound door i at iteration t, and Ω_{jt}^R the subset of feasible routes for outbound door j at iteration t. The RMP^t can be stated as follows:

$$(RMP^t)\ \text{minimize} \qquad \sum_{i\in I}\sum_{p\in\Omega_{it}^P} C_P^p\theta^p + \sum_{j\in J}\sum_{r\in\Omega_{jt}^R} C_R^r\lambda^r$$

$$\text{subject to} \quad \sum_{i\in I}\sum_{p\in\Omega_{it}^P} a_m^p\theta^p = 1 \qquad m\in M \tag{24}$$

$$\sum_{p\in\Omega_{it}^P} \theta^p \le 1 \qquad i\in I \tag{25}$$

$$\sum_{i\in I}\sum_{p\in\Omega_{it}^P} h_{jk}^p\theta^p - \sum_{r\in\Omega_{jt}^R} b_{d(k)}^r\lambda^r = 0 \qquad j\in J, k\in K \tag{26}$$

$$\sum_{j\in J}\sum_{r\in\Omega_{jt}^R} b_n^r\lambda^r = 1 \qquad n\in N \tag{27}$$

$$\sum_{r\in\Omega_{jt}^R}\left(\sum_{n\in N} q_{k(n)}b_n^r\right)\lambda^r \le Q_j \qquad j\in J \tag{28}$$

$$\lambda^r \ge 0 \qquad j\in J, r\in\Omega_{jt}^R \tag{29}$$

$$\theta^p \ge 0 \qquad i\in I, p\in\Omega_{it}^P. \tag{30}$$

The RMP^t typically contains a small subset of variables from the original sets and thus, can be solved efficiently by using a general-purpose solver. The original MP corresponds to the RMP in which $\Omega_{it}^P = \Omega_i^P$ for every $i \in I$ and $\Omega_{jt}^R = \Omega_j^R$ for every $j \in J$.

5.2 The Pricing Problem

The MP contains two huge sets of variables, $(\Omega_i^P)_{i\in I}$ and $(\Omega_j^R)_{j\in J}$. To find variables of negative reduced cost to add to the current RMP, or determine that the current solution is also optimal for MP, two pricing problems are thus necessary. The first one focuses on pricing-out assignment patterns of origins to inbound doors, whereas the second one focuses on pricing-out routes for outbound doors. Let $(\alpha, \mu, \gamma, \beta, \pi)$ be the vector of dual variables associated with constraints (24)-(28) of appropriate dimension. The reduced cost associated with assignment pattern p is

$$\overline{C}_P^p = C_P^p - \sum_{m\in M}\alpha_m a_m^p - \mu_{i(p)} - \sum_{j\in J}\sum_{k\in K}\gamma_{jk}h_{jk}^p \tag{31}$$

$$= \sum_{j\in J}\sum_{k\in K} q_k C_{i(p)j} h_{jk}^p - \mu_{i(p)} - \sum_{m\in M}\alpha_m a_m^p - \sum_{j\in J}\sum_{k\in K}\gamma_{jk}h_{jk}^p \tag{32}$$

$$= \sum_{j\in J}\sum_{k\in K} h_{jk}^p(q_k C_{i(p)j} - \gamma_{jk}) - \sum_{m\in M}\alpha_m a_m^p - \mu_{i(p)}. \tag{33}$$

Similarly, the reduced cost coefficient associated with a route r is

$$\overline{C}_R^r = C_R^r - \sum_{n\in N}\beta_n b_{nj}^r - \sum_{n\in N}\pi_j q_{k(n)} b_{nj}^r + \sum_{k\in K}\gamma_{jk}b_{d(k)j}^r \tag{34}$$

$$= \sum_{(a,b)\in A_3\cup A_4} T_{ab}g_{ab}^r - \sum_{n\in N}\beta_n b_{nj}^r - \sum_{n\in N}\pi_j q_{k(n)} b_{nj}^r + \sum_{k\in K}\gamma_{jk}b_{d(k)j}^r. \tag{35}$$

From (31)-(35), we note that the PP corresponds to the solution of two families of independent subproblems, one for the variables associated with the assignments of origins to inbound doors and the routing of commodities inside

the cross-dock, and another one for the variables associated with the routing of commodities between outbound doors and destinations. Both pricing subproblems can be in turn, further decomposed into sevral independent suproblems, one for each inbound and outbound door. In particular, for each $i \in I$, the corresponding *assignment subproblem* should be able to identify assignments with a cost structure given in (31)-(33) such that a subset of origins are assigned to inbound door i, while respecting the capacity constraint of the door, and such that each commodity associated with this subset of origins is routed to exactly one outbound door, at minimum cost. On the other hand, for each $j \in J$ the *routing subproblem* should be able to generate routes with the cost structure of (34)-(35) such that the vehicle leaves the cross-dock from door j, visits a subset of customers without exceeding its capacity, and comes back to cross-dock at door j, at minimum cost.

For each $i \in I$, given an optimal dual vector $(\alpha^t, \mu^t, \gamma^t, \beta^t, \pi^t)$ of the RMP^t, the assignment subproblem can be stated as the following integer program:

$$\text{minimize} \sum_{j \in J} \sum_{k \in K} h_{jk}(q_k C_{i(p)j} - \gamma^t_{jk}) - \sum_{m \in M} \alpha^t_m a_m - \mu^t_{i(p)} \tag{36}$$

$$\text{subject to} \quad \sum_{m \in M} O_m a_m \leq Q_i \tag{37}$$

$$\sum_{j \in J} h_{jk} - a_{o(k)} = 0 \qquad k \in K \tag{38}$$

$$a_m \in \{0,1\} \qquad m \in M \tag{39}$$

$$h_{jk} \in \{0,1\} \qquad j \in J, k \in K. \tag{40}$$

No that a_m and h_{jk} are now decision variables in the pricing problem. Constraints (37) ensure that the assignment pattern in feasible with respect to the capacity constraints. (38) ensures that if the origin of commodity k is assigned to door i, then k must be transfered from inbound door i to an outbound door j. Constraints (39) and (40) impose integrality conditions on the decision variables.

We now define $\delta(S) = \{e \in E_3 \cup E_4 : e \text{ has exactly one endpoint in } S\}$, for each $S \subseteq N \cup J$. Let g_e be an integer variable equal to the number of times that a route uses edge e along its path. Let us denote $g(\delta(S)) = \sum_{e \in \delta(S)} g_e$. For each $j \in J$, given an optimal dual vector $(\alpha^t, \mu^t, \gamma^t, \beta^t, \pi^t)$ of the RMP^t, the routing subproblem can be stated as the following integer program:

$$\text{minimize} \sum_{(a,b) \in A_3 \cup A_4} \left[T_{ab} - \left(\frac{\beta^t_a + \beta^t_b}{2} \right) - \left(\frac{D_a + D_b}{2} \right) \pi^t_j \right] g_{ab} + \sum_{k \in K} \sum_{k' \in K} \left(\frac{\gamma^t_{jk} + \gamma^t_{jk'}}{2} \right) g_{d(k)d(k')}$$

$$\text{subject to} \quad g(\delta(\{j\}) = 2 \tag{41}$$

$$g(\delta(\{a\}) = 2y_a \qquad a \in N \tag{42}$$

$$g(\delta(S)) \geq 2y_a \qquad S \subseteq N, |S| \geq 2, a \in S \tag{43}$$

$$\sum_{a \in N} D_a y_a \leq Q \tag{44}$$

$$y_a \in \{0,1\} \qquad a \in N \tag{45}$$

$$g_e \in \{0,1,2\} \qquad e \in E_3 \tag{46}$$

$$g_e \in \{0,1\} \qquad e \in E_4 \tag{47}$$

The above formulation of the capacity constrained elementary shortest path problem is inspired from the two-index vehicle-flow formulation introduced by Jepsen et al. (2008). Constraints (41) impose that exactly one route will be used. Constraints (42) impose that exactly two edges will be incident to the nodes that are actually traversed by the route, and none otherwise. Constraints (43) are the subtour elimination constraints that impose at least two edges to cross the cutset of a node subset S whenever at least one of its nodes is traversed. Constraints (44) are the capacity constraints. They impose that the total amount of load delivered by the vehicle is not larger than Q. The remaining constraints impose the binary nature of the variables, except for variables adjacent to the outbound door that can actually take a value of 2 for single-customer routes.

Our implementation of the CG algorithm starts at $t = 1$ with an empty set of assignment and routes. To ensure feasibility in the RMP^t, we add one add one artificial variables for each constraint. At each iteration t, we solve the RMP^t with the set of existing columns Ω_{it}^P and Ω_{jt}^R and obtain new dual values $(\alpha^t, \mu^t, \gamma^t, \beta^t, \pi^t)$. We then solve $|I|$ independent assignment subproblems and $|J|$ independent routing subproblems to find columns with negative reduced costs, in which case these variables are added to the RMP^{t+1} and the whole process is repeated with updated sets of columns $\Omega_{i(t+1)}^P$ and $\Omega_{j(t+1)}^R$. The CG algorithm terminates when both families of subproblems are not able to generate new variables with negative reduced costs for any of the inbound/outbound doors. By the end of the CG, we have solved the LP relaxation of F2 and thus, we obtain a valid lower bound on the optimal solution value of the DAVRP.

5.3 Solving the Pricing Problems

We next describe how the pricing subproblems (36)-(40) and (41)-(47) can be solved using specialized algorithms.

5.3.1 Solution to Assignment Subproblems

Taking into account the special structure the assignment subproblem and the fact that it does not explicitly consider the outbound door capacities, we can actually project out all routing decisions between inbound and outbound doors and transform it into a 0-1 knapsack problem in which only the assignment decisions are involved. Observe that if origin $m \in M$ is assigned to inbound door i, we can a priori determine the optimal routing between inbound and outbound doors for each commodity such that $o(k) = m$. In particular, we select the path $o(k) - i - j$ having the smallest cost $q_k C_{ij} - \gamma_{jk}^t$. That is, if $a_m = 1$ in problem (36)-(40), then the optimal route of each commodity $k \in K$,

such that $o(k) = m$, is obtained by identifying the outbound door $j(k)$ that solves the following problem:

$$j(k) \in \arg\min\{q_k C_{ij} - \gamma_{jk}^t : j \in J\}, \tag{48}$$

and by setting $h_{j(k)k} = 1$, $h_{jk} = 0$, for every $j \in J \setminus \{j(k)\}$. This means that we can a priori determine the best outbound door for each commodity in case its origin is assigned to a particular inbound door and thus, we can completely eliminate the h_{jk} variables for all assignment subproblems. For each $i \in I$, the subproblem can thus be reformulated as:

$$\text{minimize} \sum_{m \in M} \left(\sum_{k \in K : o(k) = m} \left(q_k C_{ij(k)} - \gamma_{j(k)k}^t \right) - \alpha_m^t \right) a_m - \mu_i^t \tag{49}$$

$$\text{subject to} \sum_{m \in M} O_m a_m \leq Q_i \tag{50}$$

$$a_m \in \{0, 1\} \qquad \forall m \in M. \tag{51}$$

This is a 0-1 knapsack problem which, although is known to be NP-hard, can be efficiently solved in practice by using the COMBO algorithm introduced by Martello et al. (1999).

5.3.2 Solution to Routing Subproblems

For each outbound door $j \in J$, the routing subproblem corresponds to the solution of an ESPPRC where a single resource is used to impose the vehicle capacity, besides the obvious elementarity constraints. The ESPPRC is not only known to be NP-hard but is also difficult to solve in practice. Baldacci et al. (2011) present a relaxation of the ESPPRC called the ng-route relaxation (ng-SPPRC). This relaxation aims at balancing the trade-off between the CPU time and the quality of the lower bounds obtained by relaxing the elementarity of the paths. That is, by also considering paths visiting customers more than once. It has been shown that this relaxation provides strong lower bounds while greatly decreasing the CPU times. We use such ng-route relaxation in our implementation of the solution to the routing subproblems to efficiently generate routes. For details of the definition of the ng-SPPRC and the dynamic programming algorithm used to solve it, the reader is referred to the aforementioned article.

Pecin et al. (2013) present an efficient implementation of the ng-SPPRC based on dynamic programming. We have adopted their implementation with ng-sets containing the 8 nearest customers. Note that the routes created at each iteration of the routing pricing scheme may include cycles. Pricing on non-elementary paths (routes with cycles) increases the feasible region, and thus provide weaker lower bounds of the MP as compared to pricing on elementary routes (routes without cycles). The literature suggests that this relaxation can significantly reduce CPU times while providing good bounds.

5.4 Algorithmic Refinements

At every iteration, the performance of the CG algorithm relies on the solution to the RMP^t, the knapsack problems for each inbound door, and the ng-SPPRC

for each outbound door. Preliminary computational experiments showed that the dynamic programming algorithm for the ng-SPPRC was the bottleneck of our CG algorithm. As the size of the instances increase (especially for large vehicle capacities), the time spent in the dynamic program substantially increases. We thus propose two simple procedures in order to enhance the performance of our CG algorithm.

The first procedure is related to the solution of the routing subproblems. At each iteration t, we solve the RMP^t, update the coefficients of the objective function of the ng-SPPRC associated with the first outbound door, and solve it using dynamic programming. If we are able to obtain a route with a negative reduced cost, we add it to the RMP^t and check whether this route has a negative reduced cost for the rest of the outbound doors. If so, for each outbound door giving a negative reduce cost we add the same route to the RMP^t and we do not longer solve their associated pricing problems. If we are not able to add the route for a particular door, we then solve its associated ng-SPPRC. By following this simple procedure, we typically avoid the solution of several routing subproblems per iteration. It is worth mentioning that this can be seen as a heuristic procedure for solving these problems, as the route with the minimum reduced cost coefficient is not computed for each outbound door at every iteration.

The second approach considers the identification of a promising set of initial columns to add to the RMP. This approach does not affect the performance of the subproblems but greatly reduces the number of iterations needed to obtain the optimal solution of the MP. We implement a simple local search heuristic in order to find such initial columns. We first start by creating an arbitrary feasible solution to the problem and adding its associated columns' assignment patterns and routes to the RMP. We then apply a 2-exchange operation only on the routing part. This procedure allows the exchange of two destination nodes regardless of which route they belong to. If these two customers belong to different routes associated with the same outbound door, then the assignment part still stays feasible and we add the associated two new routes to the RMP. If the customers belong to the same route, we only add a single route to the RMP. If these two customers belong to different routes associated with different outbound doors, we modify the assignment part of the problem. Fixing the assignment part only requires one to change the outbound door assignment for the commodities destined for these customers. In the end, we add the two new routes and the assignments that have been changed to the RMP. This procedure continues until we are not able to find better routes with the 2-exchange operator.

6 A Heuristic Algorithm

We next present a three-step heuristic algorithm to obtain feasible solutions to the DAVRP. In the first step, a constructive procedure that exploits the information generated during our CG algorithm is used to construct an initial feasible solution. The second step uses a simple local search with classical shift, swap, and reinsertion neighborhoods to improve the initial solution. Finally, the third step uses a more sophisticated local search that efficiently explores four very large neighborhoods with the aid of a general purpose solver to improve

the solution obtained in the second step.

6.1 Constructive Procedure

The constructive procedure consists of three phases. The first one focuses on finding a feasible solution to the routing subproblem. In the second phase, we use this information to find a promising assignment pattern for the inbound trucks to strip doors. In the third and last phase, we solve a restricted integer problem that contains the solution found at the first two stages plus some other vehicle routes and assignment patterns.

In the first phase, we use a greedy algorithm to construct routes based on the information obtained from the columns of the RMP. In particular, when the LP relaxation of $F2$ has been optimally solved, we sort the edges $(n, n') \in N \times N$ in a non-decreasing manner with respect to the number of times an edge appeared in all columns that were generated by CG. The greedy algorithm starts by selecting the most frequent available edge leaving the first outbound door. At each iteration, we look for the best ranked unselected edge incident to the previous selected edge and extend the current route until no more destination nodes can be visited that satisfy vehicle and outbound door capacities. When no more destinations can be added to a route, we complete the route by returning the vehicle to its associated door from the last visited node. We then try to create another route for a new vehicle for the same outbound door. If we fail to generate a new route due to the capacity of the outbound door, we move to the next outbound door and create a new set of routes using the same procedure. The first phase terminates once an initial feasible solution for the routing subproblem has been identified.

The second phase finds an assignment of inbound trucks to strip doors associated with the current feasible solution to the routing subproblem by solving a *generalized assignment problem* (GAP). We note that the assignment of destination nodes to outbound doors is given by the generated vehicle routes which in turn, provides the outbound door $j(k) \in J$ that commodity $k \in K$ is assigned to. The assignment subproblem can thus be solved with the following GAP:

$$\text{minimize} \sum_{m \in M} \sum_{i \in I} \sum_{k \in K: o(k) = m} q_k C_{ij(k)} X_{mi}$$

$$\text{subject to} \quad \sum_{i \in I} X_{mi} = 1 \qquad \forall m \in M \tag{52}$$

$$\sum_{m \in M} O_m X_{mi} \leq Q_i \qquad \forall i \in I \tag{53}$$

$$X_{mi} \in \{0, 1\} \qquad \forall m \in M, i \in I. \tag{54}$$

The third phase relies on optimally solving a restricted version of $F2$, where only a subset of columns are considered. In particular, once the LP relaxation of $F2$ has been optimally solved by CG, the first two phases generate an integer feasible solution using the information of the last RMP^t. We then add this integer feasible solution to the RMP^t, impose integrality conditions on both assignment and route variables of RMP^{t+1}, and solve the resulting *integer problem* without generating new columns along the enumeration tree. Since

we only have a small subset of variables, this integer program can be efficiently solved by using a general purpose solver. By the end of this constructive phase, we obtain an integer feasible solution to the DAVRP.

6.2 Local Search 1

The first local search, denoted as LS1, is based on a *Variable Neighborhood Decent* (VND) method. To the best of our knowledge, the VND method was initially proposed by Brimberg and Mladenovic (1996) and is based on a systematic search of a set of k neighborhoods, $\mathcal{N}_1, \mathcal{N}2, \cdots, \mathcal{N}_k$. In the VND, a local search in a neighborhood $\mathcal{N}_1$ is performed until a local optimal solution is found. After that, it switches to neighborhoods $\mathcal{N}_2, \cdots, \mathcal{N}_k$, sequentially, until an improved solution is found. Each time the algorithm improves the incumbent solution, it restarts the search using neighborhood $\mathcal{N}_1$. Our implementation of the VND algorithm explores three neighborhood structures. The first two focuses on modifying the inbound assignments whereas the third one modifies the routing subproblem.

$\mathcal{N}_1^{LS1}$ is a *shift* neighborhood which considers the reassignment of a single origin from a currently assigned inbound door to another, while respecting the capacity constraints. Consider an origin point m assigned to an inbound door i. We temporary assign m to a different inbound door i' such that this new assignment will remain feasible for the capacity of inbound door i'. We then change the inbound door assignment of all the commodities originating at m from i to i'. For each $m \in M$, we explore all the possible $(i, i') \in I \times I$ pairs and perform a move if the best solution improves the incumbent.

$\mathcal{N}_2^{LS1}$ is a *swap* neighborhood which considers the reassignment of two origins by interchanging their inbound doors. Consider two origin points m and m' assigned to the inbound doors i and i', respectively, such that $i \neq i'$. We reassign m to i' and m' to i, if the capacities for inbound doors i and i' are not violated by these reassignments. As a consequence, we also change the inbound door assignments of all the commodities originated at m to i' and all the commodities originated at m' to i. We explore all the feasible pairs $(m, m') \in M \times M$ and perform a move if the best solution improves the incumbent.

$\mathcal{N}_3^{LS1}$ is a *reinsertion* neighborhood that performs modifications to the routing part of the solution. It consists of removing a customer from a route and inserting it to another route. Consider a route r associated with an outbound door j that travels from a destination point n_1 to n and then from n to n_2. We remove the vertex n from route r and insert it to another route r' associated with an outbound door j', in between customer n_1' and n_2', such that $r \neq r'$, and the insertion does not violate the vehicle capacity of route r' as well as the capacity of outbound door j'. If $j = j'$, the routing between inbound and outbound doors does not change. However, if $j \neq j'$ then the change on the inner routing part is such that commodities destined for vertex n are moved from outbound door j to j'. We explore all feasible reinsertions of destinations $n \in N$ to every position of the set of existing routes, and perform a move if the best solution improves the incumbent.

6.3 Local Search 2

The second local search, denoted as LS2, is also based on a VND and focuses on exploring four more complex, very large neighborhoods, using a general purpose solver. The distinguishing feature of these new neighborhoods is that they optimally solve an assignment subproblem as part of a move evaluation each time there is a reassignment of destination nodes to outbound doors. That is, they consider simultaneous changes in both the inbound assignments and routes in every move.This is done by solving a GAP as explained in Section 6.1. As a result, performing a search on these neighborhoods is more computationally expensive as compared to LS1. However, as it will be shown in Section 7, the substantial improvement in the quality of the obtained solutions justifies their use. Before exploring the following four neighborhoods, we try to further improve the solution obtained from LS1 by fixing the routing decisions and optimally solving the inbound assignment subproblem by solving a GAP. We take the obtained solution as the input for LS2.

$\mathcal{N}_1^{LS2}$ performs modifications to the routing part by removing a customer from a route and inserting it to another route such that their associated stack doors are not the same (i.e., $j \neq j'$). A GAP is solved as a part of the move evaluation to find the inbound assignment that minimizes the handling cost for the candidate routing solution. We explore all feasible reinsertions of destinations $n \in N$ to every position of the set of routes incident to a different door, and perform a move if the best solution improves the incumbent.

$\mathcal{N}_2^{LS2}$ considers the swapping of two destinations regardless of the routes they belong to. Let n and n' be two destination points belonging to routes r and r' associated with outbound doors j and j', respectively. Let n_1 be the preceding customer and n_2 be the successor of n on route r. Similarly, let n'_1 and n'_2 be the predecessor and successor of vertex n' on route r'. We exchange the vertices n and n' such that the resulting routes r and r' do not exceed the vehicle capacities and the outbound doors j and j' do not exceed the outbound door capacities. If $j \neq j'$, we solve a GAP to find an optimal reassignment of inbound trucks. We explore all the feasible pairs of exchange $(n, n') \in N \times N$ and perform a move if the best solution improves the incumbent.

$\mathcal{N}_3^{LS2}$ considers swapping two routes regardless of the outbound doors they belong to. Let r and r' be two routes associated with outbound doors j and j' such that $j \neq j'$, respectively. We assign route r to outbound door j' and route r' to outbound door j' such that the resulting capacities on outbound doors j and j' do not exceed the outbound door capacities. We solve a GAP to find an optimal reallocation of inbound trucks. We explore all feasible pairs $(r, r') \in R \times R$ and perform a move if the best solution improves the incumbent.

$\mathcal{N}_4^{LS2}$ considers shifting a single route to an outbound door. Let r be a route associated with an outbound door j. We shift route r from outbound door j to j' such that the resulting outbound capacity for j' does not exceed the maximum outbound door capacity. As before, we solve a GAP to find an optimal reassignment of inbound trucks. We explore all feasible pairs $(r, j) \in R \times J$ and perform a move if the best solution improves the incumbent.

7 Computational Experiments

In this section we present the results of computational experiments. We first describe the set of benchmark instances. We then provide the numerical results to analyze the computational performance and limitations of the formulations and proposed solution algorithm. Finally, we evaluate the value of integrating dock-door assignment and vehicle routing decisions and provide some managerial insights.

All experiments were run on a station with an Intel Xeon CPU E5-2637 processor at 3.50 GHz and 32 GB of RAM under Linux environment. All formulations and algorithms were coded in C and solved using the callback library of CPLEX 12.6.1. We use a traditional (deterministic) branch-and-bound solution algorithm with all CPLEX parameters set to their default values. In all experiments the maximum computing time was set to 7,200 seconds.

7.1 Instance Generation

In order to generate a set of instances for the DAVRP, we have adapted and integrated two existing benchmark instances from the cross-dock and vehicle routing literature. In particular, we used the benchmark instances of Solomon (1987), also known as Solomon benchmarks proposed for the vehicle routing problem with time windows. We also used the instances of Guignard et al. (2012) proposed for the dock-door assignment problem. Both of these datasets are known to be challenging for their respective problems.

From the Solomon benchmarks, we used instances R1, RC1, C1 and C2 with 25 and 50 customers in order to generate the (x, y)-coordinates of the destination vertices. We assigned exactly one commodity for each destination vertex such that $|N| = |K|$ and we used the customer demands from Solomon's benchmarks as commodity quantities. For each data set with $|N| = 25$ we generated DAVRP instances with $|I| = |J| = 5$ and $|M| = 10$. Similarly for each set with $|N| = 50$, we generated DAVRP instances with $|I| = |J| = 10$ and $|M| = 20$. The door capacities were generated in the same way as in Guignard et al. (2012). All inbound doors have the same capacity, which was calculated by dividing the total volume of all commodities by the total number of inbound doors, given by the formula $Q_i = \left(\sum_{k \in K} q_k\right)/|I|$. Then, a capacity slack was added to the quotient. An identical procedure was used to generate outbound door capacities. Initial vehicle capacities were generated by subtracting a capacity slack of 30% from the door capacities, given by the formula $Q = Q_i - (Q_i \times 30)/100$. Then a capacity slack was added to the initial value while vehicle capacity was smaller than or equal to the door capacities. In our instances, we used slacks of 5%, 10% and 20%. The handling costs were generated in the same spirit as in Guignard et al. (2012). However, in the associated DAVRP instances, the handling costs C_{ij} and routing costs T_{abj} were defined as $C_{ij} = \beta + \sigma|i - j|$ and $T_{abj} = \pi d_{ab}$, respectively, where d_{ab} is the Euclidean distance between nodes a and b and $\beta, \sigma, \pi \geq 0$, are parameters used to control the contribution of each cost to the total cost in the overall objective function. The instances generated for our study use different values for these parameters so as to derive problems with different cost characteristics. Notably, parameters β, σ are used to control the influence of the door-to-door distances within the cross-dock in the total handling cost; and parameter π is a normalizing factor used to increase/decrease the routing costs.

In order to complete the input data for the DAVRP problem, commodities are assigned to origin points in such a way that each origin is the source of at least one and of at most $\lceil |K|/|M| \rceil + 1$ commodities.

Preliminary experiments showed that the instances with 25 customers or more could not be solved to optimality within two hours by the multi-commodity flow formulation. We thus generated smaller instances with $|N| = 10$ and $|N| = 15$ by considering only a subset of customers from the 25-customer Solomon benchmarks such that the subset of customers would represent the clusters of the original instance. The instances generated have been further sub categorized into small (set S), medium (set M), large-medium (set LM) and large (set L). The details of these instances are as described in Table 1. We have generated a total of 124 instances for these experiments.

Table 1: Summary of instances

SET	#INSTANCES	$\|I\|$	$\|J\|$	$\|M\|$	$\|N\|$	$\|K\|$
S	36	2	2	5	10	10
M	36	3	3	10	15	15
LM	36	5	5	10	25	25
L	16	10	10	20	50	50

7.2 Numerical Results of Formulations and Algorithms

In this section we present the results of computational experiments to asses the performance of the multi-commodity flow formulation, the column generation algorithm, and the heuristic. In Tables 2 and 3 we compare the performance —in terms of bounds and CPU times— of the column generation against the multi-commodity flow formulation. In Tables 4 and 5 we provide detailed results for the heuristic.

In Tables 2 and 3, the legend is as follows. Columns *CG* and *F1* stand for the column generation and the multi-commodity flow formulations, respectively. The column 'Instance' contains the name of instance. For every algorithm, under column 'LP' we report the percent deviation between the best upper bounds and LP bounds obtained by CG and F1 respectively. The column 'CPU' corresponds to the CPU times spent in seconds, and column 'OPT' corresponds to the percent deviation between the best lower bound obtained by F1 and the best upper bound overall found by any method. The LP relaxation gaps are computed as $LP = 100 \times (UB_{best} - LP_F)/UB_{best}$, where UB_{best} is the best upper bound value obtained and LP_F is the linear programming relaxation value obtained by CG and F1, respectively. The optimality gap is computed as $OPT = 100 \times (UB_{best} - LB_{F1})/UB_{best}$, where LB_{F1} is the best lower bound obtained by F1 in the given time limit. Whenever CPLEX is not able to solve an instance within the time limit, we instead report 'TIME'.

These two tables show that the CG always achieved a better LP relaxation bound than F1. We can also observe that the column generation achieved larger LP relaxation gaps for instances with large vehicle capacities. These results can be partially explained by the fact that the column relies on a route relaxation to approximate the ESPPRC. A total of 88 instances out of the 124 were not solved to optimality by F1 in the given time limit. In 57 out

Table 2: Column generation vs. Multi-commodity flow for sets S, M and LM

	CG		**F1**				**CG**		**F1**				**CG**		**F1**		
Instance	LP	CPU	LP	OPT	CPU	Instance	LP	CPU	LP	OPT	CPU	Instance	LP	CPU	LP	OPT	CPU
S1	2.73	0.07	9.59	0.00	120.51	M1	4.74	0.99	7.53	2.68	TIME	LM1	9.27	0.60	14.13	10.78	TIME
S2	2.53	0.09	8.19	0.00	47.37	M2	4.36	1.01	5.98	1.88	TIME	LM2	11.12	2.44	27.13	27.01	TIME
S3	5.61	0.08	21.58	0.00	84.43	M3	12.37	0.86	16.96	4.53	TIME	LM3	12.49	11.22	26.22	26.04	TIME
S4	5.65	0.06	19.81	0.00	92.37	M4	10.77	0.78	17.11	5.06	TIME	LM4	9.94	16.77	22.96	22.70	TIME
S5	5.18	0.09	16.79	0.00	64.12	M5	4.36	1.01	5.98	1.88	TIME	LM5	13.10	10.64	22.98	22.81	TIME
S6	3.84	0.07	12.44	0.00	56.49	M6	4.74	0.99	7.53	2.68	TIME	LM6	14.67	5.57	20.21	19.44	TIME
S7	4.14	1.07	10.51	0.00	7.62	M7	1.56	0.28	4.67	0.23	TIME	LM7	4.61	16.20	9.97	9.60	TIME
S8	0.00	0.30	14.99	0.00	84.43	M8	7.51	0.30	15.11	4.11	TIME	LM8	6.18	33.97	11.43	11.38	TIME
S9	4.06	0.01	13.68	0.00	33.93	M9	6.97	0.27	13.20	3.28	TIME	LM9	7.85	12.14	13.17	12.41	TIME
S10	7.32	0.42	13.34	0.00	19.69	M10	4.07	0.32	8.19	2.27	TIME	LM10	9.29	4.23	13.10	12.37	TIME
S11	0.00	1.93	13.18	0.00	140.25	M11	2.93	0.26	5.54	1.27	TIME	LM11	3.16	18.50	6.88	6.72	TIME
S12	7.47	1.07	12.70	0.00	65.21	M12	2.04	0.29	3.86	0.87	TIME	LM12	5.78	13.08	9.53	9.31	TIME
S13	2.50	0.06	12.27	0.00	10.43	M13	2.81	4.93	3.83	2.53	TIME	LM13	3.15	26.71	6.64	6.46	TIME
S14	4.62	0.19	12.19	0.00	58.54	M14	10.18	3.74	12.84	8.22	TIME	LM14	3.44	11.99	6.87	6.74	TIME
S15	0.00	0.28	10.61	0.00	84.99	M15	4.13	3.58	5.62	3.19	TIME	LM15	1.17	65.47	4.36	4.19	TIME
S16	4.14	0.54	10.51	0.00	19.39	M16	3.18	3.98	4.01	2.20	TIME	LM16	3.14	32.91	6.17	6.12	TIME
S17	6.12	0.52	10.00	0.00	13.90	M17	2.12	5.40	2.67	1.51	TIME	LM17	4.86	46.29	17.56	17.22	TIME
S18	5.73	0.33	9.75	0.00	107.33	M18	10.59	6.31	14.42	9.03	TIME	LM18	10.98	19.05	20.22	19.40	TIME
S19	0.00	0.44	9.64	0.00	290.58	M19	5.18	1.53	6.27	5.07	TIME	LM19	8.17	11.12	13.90	11.81	TIME
S20	5.19	0.23	9.46	0.00	22.12	M20	17.79	1.74	21.34	18.16	TIME	LM20	4.59	9.69	8.69	8.50	TIME
S21	1.57	1.41	9.09	0.00	2.90	M21	15.78	1.10	18.40	14.66	TIME	LM21	7.71	3.90	10.05	9.85	TIME
S22	0.00	0.27	8.88	0.00	118.18	M22	7.71	1.14	9.26	7.75	TIME	LM22	16.95	10.36	19.80	19.62	TIME
S23	4.27	0.19	8.23	0.00	36.57	M23	5.48	1.12	6.38	4.87	TIME	LM23	3.92	28.70	6.78	6.72	TIME
S24	1.65	0.58	8.11	0.00	0.31	M24	3.66	1.31	4.26	3.27	TIME	LM24	5.20	119.04	7.93	7.90	TIME
S25	0.00	0.41	8.08	0.00	149.68	M25	5.58	0.20	14.79	5.68	TIME	LM25	7.50	4.00	10.30	9.28	TIME
S26	4.61	0.01	7.53	0.00	3.78	M26	2.00	0.26	5.95	2.45	TIME	LM26	4.83	4.95	7.48	7.19	TIME
S27	1.87	0.40	7.49	0.00	0.35	M27	2.00	0.25	5.30	2.03	TIME	LM27	3.84	10.14	6.10	6.01	TIME
S28	4.02	0.31	7.33	0.00	21.89	M28	7.17	0.18	19.04	10.05	TIME	LM28	2.49	19.95	4.53	4.29	TIME
S29	3.91	0.73	6.65	0.00	125.40	M29	2.82	0.18	8.35	4.00	TIME	LM29	0.63	60.78	2.35	2.28	TIME
S30	3.20	0.19	6.17	0.00	5.66	M30	5.58	0.20	14.79	5.68	TIME	LM30	11.20	30.24	20.67	20.53	TIME
S31	2.11	0.18	5.36	0.00	27.26	M31	8.47	0.36	14.60	4.86	TIME	LM31	16.20	3.85	21.99	20.54	TIME
S32	1.98	0.01	5.13	0.00	3.24	M32	4.96	0.24	9.04	4.75	TIME	LM32	4.56	29.27	9.60	9.55	TIME
S33	2.04	0.02	4.73	0.00	99.74	M33	3.42	0.28	6.48	1.82	TIME	LM33	14.19	0.60	19.18	16.72	TIME
S34	1.37	0.46	4.53	0.00	20.45	M34	3.59	0.26	6.18	1.89	TIME	LM34	1.72	62.84	6.43	6.34	TIME
S35	1.91	0.68	4.43	0.00	7.79	M35	2.51	0.29	4.32	1.41	TIME	LM35	5.02	29.90	8.62	7.46	TIME
S36	1.06	1.10	3.51	0.00	0.23	M36	8.43	0.29	15.96	3.78	TIME	LM36	9.23	5.56	12.77	12.28	TIME
Average	3.12		9.90	0.00			5.88		9.60	4.43			7.28		12.69	12.15	

Table 3: Column generation vs. Multi-commodity flow for set L

	CG		**F1**		
Instance	LP	CPU	LP	OPT	CPU
L1	15.27	1272.57	22.87	21.58	TIME
L2	16.55	535.38	25.86	24.32	TIME
L3	13.12	530.38	26.32	23.22	TIME
L4	12.82	411.86	15.49	13.54	TIME
L5	13.99	897.00	21.71	16.10	TIME
L6	13.23	421.52	22.92	18.41	TIME
L7	14.61	512.14	22.94	19.67	TIME
L8	7.44	357.13	10.27	5.05	TIME
L9	6.87	1163.09	15.36	9.47	TIME
L10	6.53	517.16	17.21	11.35	TIME
L11	9.48	785.94	20.60	11.07	TIME
L12	5.08	449.69	8.92	4.76	TIME
L13	7.57	1278.53	16.00	13.79	TIME
L14	8.77	543.62	19.19	14.75	TIME
L15	12.85	604.24	25.10	13.87	TIME
L16	19.02	802.18	25.22	17.46	TIME
Average	11.45		19.75	14.90	

of these 88 instances, the LP relaxation bound obtained by CG was tighter than the best lower bound obtained by F1 in the given time limit. In Tables 4 and 5 we report detailed results for the heuristic. In order to evaluate the benefit of each of the steps of our heuristic we report the %deviation of the solution obtained at the end of each step. In these tables, 'CH' stands for the constructive heuristic. 'LS1' corresponds to the results obtained by the first local search, and 'LS2' corresponds to the results obtained with second local search. Under column labeled 'F1' we report the solutions obtained by the the multi-commodity flow formulation within the time limit of two hours, whenever this was possible. For the instances given in Table 4, the CPU times for LS1 never exceeded a second and the overall CPU times never exceeded 150 seconds consistently throughout all the instances. For this reason, the CPU times are omitted. Columns 'UB' correspond to the percent deviation between the upper bound obtained by heuristics or the multicommodity flow formulation and best upper bounds obtained. Upper bound gaps are computed as $UB = 100 \times (UB_H - UB_{best})/UB_H$, where UB_{best} is the best solution value obtained and UB_H is the solution value obtained by CH, LS1, LS2, and F1. Column '(%)GAP' refers to the percent deviation between the lower bound obtained by column generation and the best upper bound obtained by the heuristics such that, $(\%)GAP = 100 \times (UB_{BEST} - CG_{LP})/UB_{BEST}$, where UB_{BEST} is the best upper bound obtained by any of the heuristics and CG_{LP} is the lower bound obtained by column generation. As shown in these two tables, CPLEX is not capable to find a feasible solution for 4 instances in the set LM and for all of the instances in L set in the given time limit. For these instances we report 'N/A' for the 'UB' column under 'F1'.

The heuristic was able to provide the optimal solution for all the instances in set S. For the instances for which the optimality could not be proven by

Table 4: Detailed results for the heuristic on sets S, M and LM

	CH	**LS1**	**LS2**		**F1**		**CH**	**LS1**	**LS2**		**F1**		**CH**	**LS1**	**LS2**		**F1**
Instance	UB	UB	UB	(%)GAP	UB	Instance	UB	UB	UB	(%)GAP	UB	Instance	UB	UB	UB	(%)GAP	UB
S1	0.00	0.00	0.00	2.73	0.00	M1	2.19	2.19	0.00	4.74	1.33	LM1	0.78	0.78	0.00	9.27	42.19
S2	0.00	0.00	0.00	2.53	0.00	M2	1.23	1.23	0.00	4.36	0.13	LM2	8.62	0.22	0.00	11.12	26.47
S3	0.00	0.00	0.00	5.61	0.00	M3	3.32	3.32	0.00	12.37	0.00	LM3	15.78	8.64	0.00	12.49	26.52
S4	0.00	0.00	0.00	5.65	0.00	M4	4.76	4.76	0.00	10.77	3.85	LM4	7.38	1.89	0.00	9.94	10.13
S5	0.00	0.00	0.00	5.18	0.00	M5	2.60	2.60	0.00	4.36	0.13	LM5	8.97	0.93	0.00	13.10	22.23
S6	0.00	0.00	0.00	3.84	0.00	M6	2.72	2.72	0.00	4.74	1.33	LM6	14.31	0.00	0.00	14.67	12.16
S7	0.00	0.00	0.00	4.14	0.00	M7	0.10	0.10	0.00	1.56	1.50	LM7	2.91	0.51	0.00	4.61	6.93
S8	0.00	0.00	0.00	0.00	0.00	M8	1.01	0.00	0.00	7.51	0.00	LM8	3.38	0.37	0.00	6.18	2.49
S9	0.00	0.00	0.00	4.06	0.00	M9	4.95	4.59	0.56	6.97	0.00	LM9	7.01	0.00	0.00	7.85	1.75
S10	0.75	0.75	0.00	7.32	0.00	M10	0.55	0.55	0.55	4.07	0.00	LM10	10.32	0.00	0.00	9.29	14.16
S11	0.00	0.00	0.00	0.00	0.00	M11	2.53	0.00	0.00	2.93	0.00	LM11	2.38	0.58	0.00	3.16	4.58
S12	0.00	0.00	0.00	7.47	0.00	M12	0.16	0.16	0.16	2.04	0.00	LM12	4.01	0.24	0.00	5.78	1.84
S13	3.04	0.00	0.00	2.50	0.00	M13	1.03	1.03	0.00	2.81	0.37	LM13	8.15	1.04	0.00	3.15	1.74
S14	0.00	0.00	0.00	4.62	0.00	M14	2.24	0.54	0.54	10.18	0.00	LM14	4.41	0.81	0.00	3.44	7.67
S15	0.00	0.00	0.00	0.00	0.00	M15	4.53	2.65	0.91	5.01	0.00	LM15	2.09	0.42	0.00	1.17	5.21
S16	0.00	0.00	0.00	4.14	0.00	M16	1.02	0.86	0.00	3.18	0.00	LM16	0.86	0.58	0.00	3.14	2.49
S17	0.00	0.00	0.00	6.12	0.00	M17	1.09	1.09	0.46	2.12	0.00	LM17	5.91	1.68	0.00	4.86	4.62
S18	0.00	0.00	0.00	5.73	0.00	M18	3.78	3.78	0.00	10.59	1.33	LM18	4.82	1.29	0.00	10.98	2.61
S19	0.00	0.00	0.00	0.00	0.00	M19	1.44	1.43	0.00	5.18	0.25	LM19	4.53	0.35	0.00	8.17	6.18
S20	0.53	0.53	0.00	5.19	0.00	M20	2.57	1.23	0.00	17.79	0.00	LM20	4.19	0.28	0.00	4.59	3.04
S21	7.87	0.81	0.00	1.57	0.00	M21	5.16	0.00	0.00	15.78	0.00	LM21	4.85	0.38	0.00	7.71	N/A
S22	0.00	0.00	0.00	0.00	0.00	M22	3.53	0.53	0.53	7.71	0.00	LM22	19.16	2.16	0.00	16.95	0.00
S23	0.18	0.00	0.00	4.27	0.00	M23	3.12	2.25	0.00	5.48	0.00	LM23	5.73	0.40	0.00	3.92	1.58
S24	0.00	0.00	0.00	1.65	0.00	M24	0.64	0.64	0.59	3.67	0.00	LM24	4.94	0.00	0.00	5.20	0.00
S25	0.00	0.00	0.00	0.00	0.00	M25	5.48	5.48	0.00	5.58	0.00	LM25	2.05	1.90	0.00	7.50	N/A
S26	3.04	0.41	0.00	4.61	0.00	M26	1.32	1.32	0.00	2.00	0.00	LM26	8.02	1.20	0.00	4.83	3.44
S27	0.00	0.00	0.00	1.87	0.00	M27	1.41	1.41	0.83	2.00	0.00	LM27	1.60	0.18	0.00	3.84	2.63
S28	0.41	0.41	0.00	4.02	0.00	M28	6.23	3.11	0.00	7.17	0.00	LM28	1.02	0.20	0.00	2.49	0.94
S29	0.00	0.00	0.00	3.91	0.00	M29	3.66	0.84	0.11	2.82	0.00	LM29	1.13	0.23	0.00	0.63	2.27
S30	0.00	0.00	0.00	3.20	0.00	M30	5.48	5.48	0.00	5.58	0.00	LM30	3.91	1.69	0.00	11.20	6.27
S31	0.00	0.00	0.00	2.11	0.00	M31	5.70	5.70	0.00	8.47	0.00	LM31	13.33	4.30	0.00	16.20	N/A
S32	8.37	1.96	0.00	1.98	0.00	M32	6.61	1.46	0.00	4.96	0.00	LM32	8.63	2.51	0.00	4.56	0.08
S33	0.00	0.00	0.00	2.04	0.00	M33	1.81	0.62	0.00	3.42	0.00	LM33	1.50	0.76	0.00	14.19	N/A
S34	6.71	4.33	0.00	1.37	0.00	M34	3.68	2.25	0.00	3.59	0.00	LM34	3.05	0.62	0.00	1.72	3.25
S35	0.09	0.09	0.00	1.91	0.00	M35	1.84	0.88	0.00	2.51	0.00	LM35	1.60	0.26	0.00	5.02	2.08
S36	4.47	0.00	0.00	1.06	0.00	M36	1.52	1.52	0.00	8.43	0.00	LM36	8.43	0.22	0.00	9.23	5.36
Average	0.98	0.26	0.00	3.12	0.00		2.81	1.90	0.15	5.90	0.28		5.83	1.07	0.03	7.28	N/A
# Best	26	29	36		36		0	3	26		27		0	4	36		2

Table 5: Detailed results for the heuristic on set L

	CH		**LS1**	**LS2**			**F1**
Instance	UB	CPU	UB	UB	CPU	(%)GAP	UB
L1	8.75	2316.71	5.87	0.00	2356.00	15.27	N/A
L2	14.90	1625.90	4.05	0.00	1639.92	16.55	N/A
L3	31.93	1690.75	3.17	0.00	1825.90	13.12	N/A
L4	16.55	715.54	13.99	0.00	759.16	12.82	N/A
L5	22.73	1942.32	1.63	0.00	1989.78	13.99	N/A
L6	10.70	1525.79	10.70	0.00	1568.76	13.23	N/A
L7	14.92	1582.97	6.51	0.00	1613.78	14.61	N/A
L8	10.76	469.72	6.07	0.00	532.94	7.44	N/A
L9	12.98	2204.85	7.35	0.00	2263.63	6.87	N/A
L10	18.71	1636.87	10.19	0.00	1783.03	6.53	N/A
L11	18.40	1871.00	2.80	0.00	2010.58	9.48	N/A
L12	7.21	1569.81	3.45	0.00	1643.17	5.08	N/A
L13	16.85	2382.85	11.54	0.00	2415.10	7.57	N/A
L14	29.72	1563.57	0.53	0.00	1589.89	8.77	N/A
L15	24.46	1755.39	7.28	0.00	1963.83	12.85	N/A
L16	12.19	1060.84	4.07	0.00	1291.31	19.02	N/A
Average	16.99		6.20	0.00		11.45	N/A
# Best	0		0	16			0

F1 in the given time limit, the heuristic provides better upper bounds overall. Formulation F1 obtained better upper bounds than the heuristic for 10 instances in set M and for only one instance in set LM. For the set L, F1 could not find a feasible solution for all of the instances within two hours. For 10 instances in the set M, 34 instances in set LM and for all the instances in set L, the heuristic provided better upper bounds than those obtained by F1. Overall, the heuristic is capable to provide the best upper bounds on 60 instances. Table 5 shows that the CPU times increased dramatically for CH as the size of the instance increased. These CPU times can be explained by the number of iterations needed for the column generation to converge and the difficulty of solving the pricing subproblems at each time. Also, as the size of the instances get larger, the CPU times increase for LS2 because of the GAP's that need to be solved to evaluate the different neighborhoods of the local search. Although LS1 and CH performed well for instances in Table 4, it was LS2 that provides a substantial gain in the solution quality for the instances in the set L.

7.3 The Value of Integration

In this section we evaluate the added value of using an integrated solution approach rather than a sequential one. In particular, we seek to find out how much of the total cost we save by integrating dock-door assignment and vehicle routing in a cross-docking context. When transportation costs are dominant in terms of total cost, one might employ a sequential solution strategy in which the routes are designed first and dock door assignment decisions are optimized based on these routes. Similarly, when handling cost are dominant one may use a sequential approach in which dock door decisions are optimized first and routing decisions are optimized after based on the selected inbound assignments. We perform a set of computational experiments to directly compare our integrated approach versus these two sequential solution approaches:

- *Sequential approach 1* (SA1): In the first stage we first optimize the routing decisions by ignoring the handling costs. This is done by solving a *capacitated vehicle routing problem* associated with set of destinations N. In the second stage we solve a CDAP in which every route from the first stage represents a destination node. In this way, the routes are assigned to their best possible outbound door with respect to the handling cost, while respecting the door capacities.

- *Sequential approach 2* (SA2): In the first stage we first optimize the dock door assignments by disregarding the routing cost. This is done by solving a CDAP in which each destination point in N has to be directly assigned to a single outbound door. In the second stage we solve a capacitated vehicle routing problem for each outbound door using the assignment of commodities to outbound doors given in the first stage.

We have generated a set of DAVRP instances each containing three inbound and outbound doors, four origin nodes, 10 destinations and 10 different commodities. We use parameters $\beta = 4, \sigma = 4, \pi = 1$, and we consider a new parameter $\eta \in [0, 1]$ such that each handling cost C_{ij} is multiplied by η and each routing cost by $(1 - \eta)$. The parameters β, σ, π are chosen in such a way that a value of $\eta = 0.5$ roughly matches with a 50% of the total cost that comes from the handling. For these experiments, we use a total of 81 instances in which the only difference between them is the value of η which ranges from 0 to 1 in increments of 0.0125, i.e, $\eta \in \{0, 0.0125, 0.025, \cdots, 0.975, 0.9875, 1\}$.

In Figure 3 we plot the percent deviation of employing a sequential approach, as compared with our integrated approach. In the x-axis we denote the percent of the total cost that is due to the handling operations. In the y-axis, we report the percent deviation of the sequential approach as compared to an integrated one, as follows. For each problem, the two sequential approaches have been executed and their optimal solutions found with the help of our flow formulation. The one attaining the best result (in terms of total cost) is retained and its deviation is reported. If we denote this cost by z_{seq} and let z_{int} denote the total cost attained by employing an integrated approach, the percent deviation is computed as $dev = 100 \times (z_{seq} - z_{int})/z_{int}$.

According to these results, the decision maker can save up to a 12% of the total cost by adopting an integrated approach. Moreover, even when the routing costs are dominant (Handling (%) $<$ 30%), an integrated approach provides savings of up to 5% with respect to a sequential approach. Similarly, when the material handing costs are dominant (Handling (%) $>$ 70%), the cost difference between integrated and sequential approach is fairly significant. Even in cases where handling costs are extremely dominant (Handling(%) $>$ 80), a sequential solution strategy provides poor results with an increment in the cost of around 2%. Similarly, in cases where the routing costs are extremely dominating (Handling(%) $<$ 20%), the decision maker could save up to a similar amount by adopting an integrated approach.

Using the same set of instances, we also report and analyze the deviation obtained by each sequential approach if applied independently, as compared to our integrated approach. Figures 4 and 5 depict the results obtained by adopting SA1 and SA2, respectively. The trend in Figure 4 shows that optimizing the routing part first results in higher cost deviations as the material handling cost

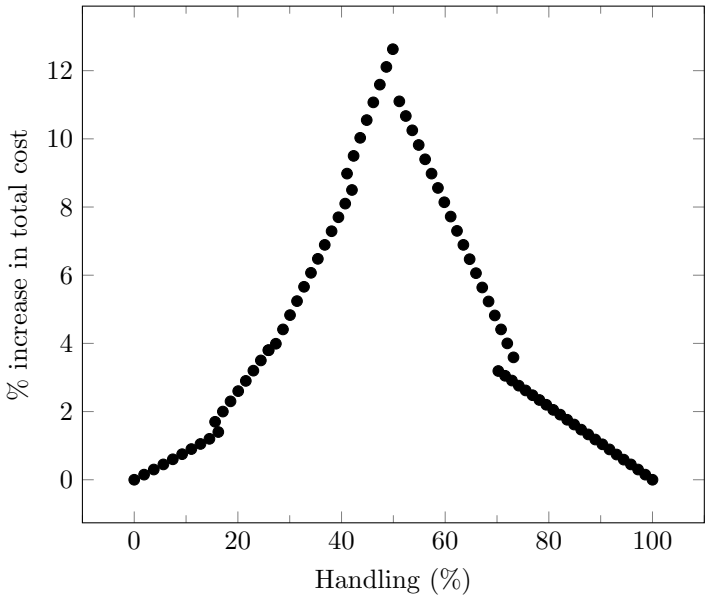

Figure 3: Value of Integration

percentage gets higher. Similarly, the trend in Figure 5 demonstrates that adopting SA2 as a solution strategy gives poor results as the percentage of routing costs gets lower. These results are similar to those obtained in Figure 3. Furthermore, both sequential approaches provide poor results for extreme cases where one of the costs is dominating. Even if either cost dominates the other by up to a 80%, the integrated approach can still provide gains of up to a 2%, and this grows to up to a 40% if the wrong sequential optimization is chosen.

8 Conclusions

In this article we introduced the *Dock-Door Assignment and Vehicle Routing Problem* (DAVRP). It is a combinatorial optimization problem combining two important decisions in a cross-docking context, namely dock-door assignment and vehicle routing. To the best of our knowledge, this problem has not been previously studied in the literature. We presented two MIP formulations for the DAVRP. We also developed a column generation algorithm based on a set-partitioning formulation and demonstrated that the resulting subproblems correspond to 0-1 knapsack problems and elementary shortest path problems under resource constraints, two NP-hard problems that have been vastly studied in the scientific literature and for which efficient algorithms exist. We have also introduced a three-phase heuristic that finds good quality solutions of the DAVRP in short computing times. Furthermore, we have conducted computational experiments to assess the impact of the new modeling framework as well as that of the different solutions strategies adopted. Notably, we have shown that an integrated solution approach may lead to significant savings in costs, even in the case in which one cost dominates the other.

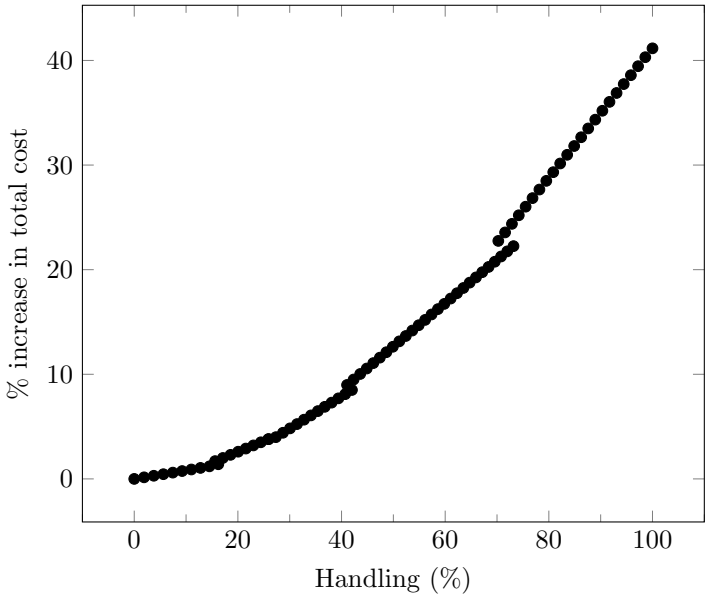

Figure 4: % Deviation of employing SA1 when compared to an integrated approach

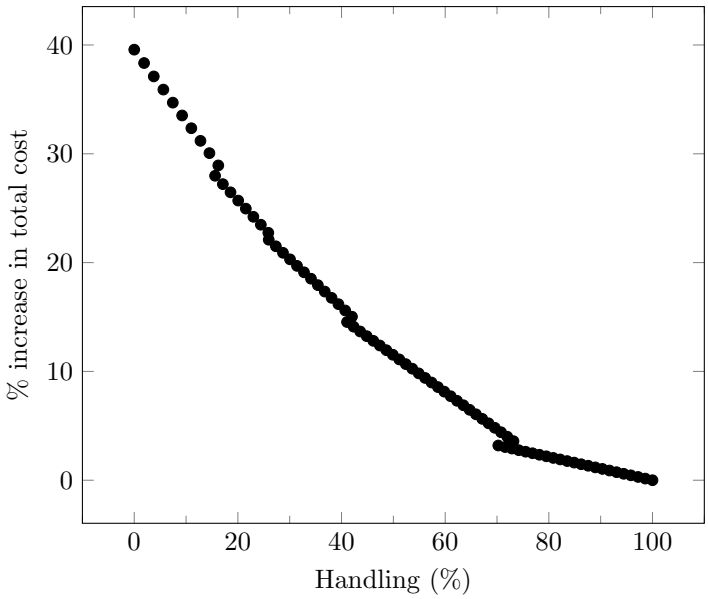

Figure 5: % Deviation of employing SA2 when compared to an integrated approach

Acknowledgments

This research has been partially funded by the Canadian Natural Sciences and Engineering Research Council of Canada (NSERC) and the *Fonds de recherche du Québec - Nature et technologies* (FRQNT). These supports are gratefully acknowledged. The authors also thank the two anonymous reviewers for their valuable comments.

References

References

Agustina, D., C. Lee, and R. Piplani (2010). A review: Mathematical models for cross docking planning. *International Journal of Engineering Business Management 2*(2), 47–54.

Agustina, D., C. Lee, and R. Piplani (2014). Vehicle scheduling and routing at a cross docking center for food supply chains. *International Journal of Production Economics*, 29–41.

Apte, U. M. and S. Viswanathan (2000). Effective cross docking for improving distribution efficiencies. *International Journal of Logistics 3*(3), 291–302.

Arnaout, G., E. Rodriguez-Velasquez, G. Rabadi, and R. Musa (2010). Modeling cross-docking operations using discrete event simulation. In *Proceedings of the 6th International Workshop on Enterprise & Organizational Modeling and Simulation*, pp. 113 – 120.

Baldacci, R., A. Mingozzi, and R. Roberti (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research 59*(5), 1269–1283.

Belle, J. V., P. Valckenaers, and D. Cattrysse (2012). Cross-docking: State of the art. *Omega 40*(6), 827–846.

Boysen, N. and M. Fliedner (2010). Cross dock scheduling: Classification, literature review and research agenda. *Omega 38*(6), 413 – 422.

Bozer, Y. A. and H. J. Carlo (2008). Optimizing inbound and outbound door assignments in less-than-truckload cross-docks. *IIE Transactions 40*(11), 1007–1018.

Brimberg, J. and N. Mladenovic (1996). A variable neighbourhood algorithm for solving the continuous location-allocation problem. *Studies in Locational Analysis 10*, 1–12.

Buijs, P., I. F. Vis, and H. J. Carlo (2014). Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research 239*(3), 593 – 608.

Chandran, P. M. (2003). Wal-mart's supply chain management practices. Technical report, ICFAI Center for Management Research.

Choy, K., H. Chow, T. Poon, and G. Ho (2012). Cross-dock job assignment problem in space-constrained industrial logistics distribution hubs with a single docking zone. *International Journal of Production Research 50*(9), 2439–2450.

Dondo, R. and J. Cerdá (2013). A sweep-heuristic based formulation for the vehicle routing problem with cross-docking. *Computers & Chemical Engineering 48*(1), 293–311.

Garvin, W. W., H. W. Crandall, J. B. John, and R. A. Spellman (1957). Applications of linear programming in the oil industry. *Management Science 3*(4), 407–430.

Guignard, M., P. M. Hahn, A. A. Pessoa, and D. C. da Silva (2012). Algorithms for the cross-dock door assignment problem. In *Proceedings of the Fourth International Workshop on Model-based Metaheuristics*.

Jepsen, M., B. Petersen, and S. Spoorendonk (2008). A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint. Technical Report 08/01, University of Copenhagen.

Kuo, Y. (2013). Optimizing truck sequencing and truck door assignment in a cross docking system. *Expert Systems with Applications 40*, 5532–5541.

Liao, C.-J., Y. Lin, and S. C. Shih. (2010). Vehicle routing with cross-docking in the supply chain. *Expert Systems with Applications 37*(10), 6868–6873.

Liao, T., P. Egbelu, and P. Chang (2013). Simultaneous dock assignment and sequencing of inbound trucks under a fixed outbound truck schedule in multi-door cross docking operations. *International Journal of Production Economics 141*, 212–229.

Luo, G. and J. S. Noble (2012). An integrated model for crossdock operations including staging. *International Journal of Production Research 50*(9), 2451–2464.

Martello, S., D. Pisinger, and P. Toth (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science 45*(3), 414–424.

Morais, V. W., G. R. Mateus, and T. F. Noronha (2014). Iterated local search heuristics for the vehicle routing problem with cross-docking. *Expert Systems with Applications 41*(16), 7495 – 7506.

Naskaris, S., R. Pudukadan, K. Smith, and M. Maher (2014). Cost analysis for a cross dock alternative to supply local produce from food hub to grocery store via a large regional distribution center. Technical report, Center for Environmental Farming Systems, North Carolina State University, Raleigh, NC.

Nassief, W., I. Contreras, and R. As'ad (2016). A mixed-integer programming formulation and lagrangean relaxation for the cross-dock door assignment problem. *International Journal of Production Research 2*(54), 494–508.

Nassief, W., I. Contreras, M. Guignard, P. Hahn, and B. Jaumard (2017). The container scheduling a cross dock-door selection problem. *submitted*.

Nassief, W., I. Contreras, and B. Jaumard (2017). A comparison of formulations and relaxations for cross-dock door assignment problems. *submitted*.

Oh, Y., H. Hwang, C. N. Cha, and S. Lee (2006). A dock-door assignment problem for the korean mail distribution center. *Computers & Industrial Engineering 51*(2), 288–296.

Pecin, D., M. Poggi, and R. Martinelli (2013). Efficient elementary and restricted non-elementary route pricing. Technical report, Pontifica Universida de Catolica do Rio De Janeiro.

Peck, K. (1983). *Operational Analysis of Freight Terminals Handling Less Than Container Load Shipments*. University of Illinois at Urbana-Champaign.

Santos, F. A., G. R. Mateus, and A. S. D. Cunha (2011a). A branch-and-price algorithm for a vehicle routing problem with cross-docking. *Electronic Notes in Discrete Mathematics 37*(1), 249–254.

Santos, F. A., G. R. Mateus, and A. S. D. Cunha (2011b). A novel column generation algorithm for the vehicle routing problem with cross-docking. In *Proceedings of the 5th international conference on Network optimization*, pp. 412–425. Springer Verlag.

Santos, F. A., G. R. Mateus, and A. S. D. Cunha (2013). The pickup and delivery problem with cross-docking. *Computers & Operations Research 40*, 1085–1093.

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

Tarantilis, C. D. (2013). Adaptive multi-restart tabu search algorithm for the vehicle routing problem with cross-docking. *Optimization Letters 7*(7), 1583–1596.

Tsui, L. Y. and C.-H. Chang (1990). A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers & Industrial Engineering 19*(1-4), 309–312.

Tsui, L. Y. and C.-H. Chang (1992). An optimal solution to a dock door assignment problem. *Computers & Industrial Engineering 23*(1-4), 283–286.

Wen, M., J. Larsen, J. Clausen, J. F. Cordeau, and G. Laporte (2009). Vehicle routing with cross-docking. *Journal of the Operational Research Society 60*(12), 1708–1718.

Young, L., J. Jung, and K. Lee. (2006). Vehicle routing scheduling for cross-docking in the supply chain. *Computers & Industrial Engineering 51*(2), 247–256.

Zhu, Y.-R., P. M. Hahn, Y. Liu, and M. Guignard-Spielberg (2009). New approach for the cross-dock door assignment problem. In *Pesquisa Operacional na Gestão do Conhecimento*.