A Matheuristic Algorithm for Solving the Vehicle Routing Problem with Cross-Docking

Aldy Gunawan[1(✉)], Audrey Tedja Widjaja[1], Pieter Vansteenwegen[2], and Vincent F. Yu[3]

[1] Singapore Management University, 80 Stamford Road, Singapore, Singapore
{aldygunawan,audreyw}@smu.edu.sg
[2] Centre for Industrial Management/Traffic and Infrastructure, KU Leuven, Celestijnenlaan 300, Box 2422, Leuven, Belgium
pieter.vansteenwegen@kuleuven.be
[3] Department of Industrial Management, National Taiwan University of Science and Technology, 43, Sec. 4, Keelung Road, Taipei 106, Taiwan
vincent@mail.ntust.edu.tw

Abstract. This paper studies the integration of the vehicle routing problem with cross-docking, namely VRPCD. The aim is to find a set of routes to deliver single products from a set of suppliers to a set of customers through a cross-dock facility, such that the operational and transportation costs are minimized, without violating the vehicle capacity and time horizon constraints. A two-phase matheuristic approach that uses the routes of the local optima of an adaptive large neighborhood search (ALNS) as columns in a set-partitioning formulation of the VRPCD is designed. This matheuristic outperforms the state-of-the-art algorithms in solving a subset of benchmark instances.

Keywords: Vehicle routing problem · Cross-docking · Scheduling · Matheuristic

1 Introduction

Cross-docking is an intermediate activity within a supply chain network for enabling a transshipment process. The purpose is to consolidate different shipments for a particular destination in a full truckload (FTL), such that direct shipment with less than truckload (LTL) can be avoided, and thus the transportation cost is minimized [1]. The VRPCD as the integration of the vehicle routing problem (VRP) and cross-docking was first introduced by [5], which aims to construct a set of routes to deliver a single type of products from a set of suppliers to a set of customers through a cross-dock facility, such that the operational and transportation costs are minimized, with respect to vehicle capacity and time limitations.

I. S. Kotsireas and P. M. Pardalos (Eds.): LION 14 2020, LNCS 12096, pp. 9–15, 2020.
https://doi.org/10.1007/978-3-030-53552-0_2

The idea of combining metaheuristics with elements of exact mathematical programming algorithms, known as matheuristics, for solving the VRP was first introduced by [3]. [4] introduced matheuristic based on large neighborhood search for solving the VRPCD with resource constraints. In this study, we design a matheuristic which only requires a heuristic scheme to generate columns [2]. The column generation scheme is performed by an adaptive large neighborhood search (ALNS) and the set partitioning formulation is used to solve a subset of columns to find the final solution. The matheuristic is tested on one set of benchmark VRPCD instances, and the results are compared against those of the state-of-the-art algorithms. Preliminary experimental results show that our proposed matheuristic is able to obtain 29 out of 30 optimal solutions and outperform the state-of-the-art algorithms: tabu search (TS) [5], improved tabu search (imp-TS) [6], and simulated annealing (SA) [7].

2 Problem Description

The VRPCD network consists of a set of suppliers $S = \{1, 2, \ldots, |S|\}$ delivering a single product to a set of customers $C = \{1, 2, \ldots, |C|\}$ through a single cross-dock facility, denoted as node 0. Two major processes involved are: the pickup process at the suppliers and the delivery process to the customers. P_i products must be picked up from node i in S, and D_i products must be delivered to node i in C. Each pair of nodes (i, j) in S is connected by travel time t'_{ij} and transportation cost c'_{ij}. Each pair of nodes (i, j) in C is connected by travel time t''_{ij} and transportation cost c''_{ij}. The VRPCD network is illustrated in Fig. 1.

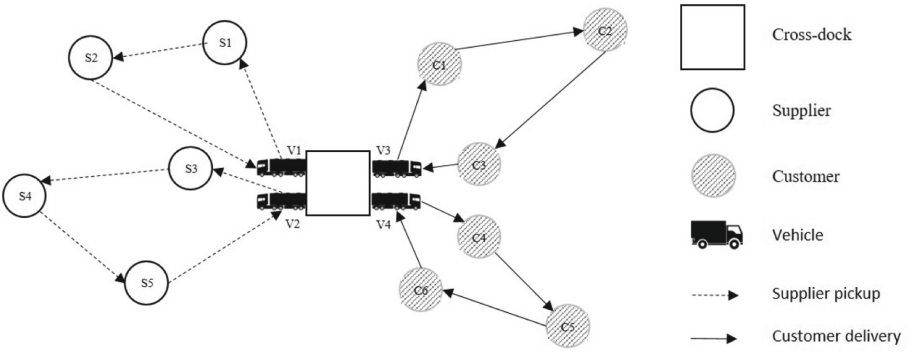

Fig. 1. VRPCD network

A fleet of homogeneous vehicles $V = \{1, 2, ..., |V|\}$ with capacity Q is available at the cross-dock facility to be utilized for shipments. Each vehicle can only perform either a pickup process or a delivery process, or neither. In the pickup process, vehicles depart from the cross-dock, visit one (or more) supplier(s) to pickup their products, and return to the cross-dock for consolidating

products. After the products are consolidated according to customers' demand, vehicles depart from the cross-dock, visit one (or more) customer(s) to deliver their demand, and return to the cross-dock. For each vehicle used, an operational cost H will be charged. The VRPCD aims to determine the number of vehicles used and its corresponding routes, such that the operational and transportation costs are minimized. The constraints in the VRPCD are as follows:

- the total transportation time for the pickup and delivery processes together does not exceed T_{max}
- each supplier and customer can only be visited exactly once
- the number of vehicles utilized in both the pickup and delivery process together does not exceed $|V|$
- the amount of loads on the pickup route and on the delivery route in each vehicle does not exceed Q.

3 Proposed Algorithm

The matheuristic is decomposed into two phases: (i) adaptive large neighborhood search (ALNS) and (ii) the set partitioning formulation. The first phase aims to generate feasible candidate routes as many as possible, represented as columns. Those routes are then accommodated in two different pools, Ω_s and Ω_c, for pickup and delivery process respectively. In the second phase, a set partitioning formulation is solved over the set of routes stored in Ω_s and Ω_c to find a combination of routes that satisfies the VRPCD constraints. We define Sol_0 and Sol^* as the current and the best found solutions so far. An initial solution is constructed based on a greedy approach, where the node with the least additional transportation cost is inserted, such that each vehicle starts (ends) its route from (in) the cross-dock without violating the vehicle capacity and time horizon constraints. First, the Sol_0 and Sol^* are set to be the same as the initial solution. Then, the constructed routes (pickup routes and delivery routes) are added into Ω_s and Ω_c respectively. Let $R = \{1, 2, \ldots, |R|\}$ be a set of destroy operators, $I = \{1, 2, \ldots, |I|\}$ be a set of repair operators. The score s_j and weight w_j of each operator $j \in R \cup I$ is set such that its probability of choosing each operator j, p_j, in both R and I is equally likely in the beginning. At each iteration, a destroy operator R_i is randomly selected to remove π nodes from Sol_0. Consequently, a repair operator I_i is selected to reinsert the π removed nodes back to the Sol_0, resulting in a new neighborhood solution. In our implementation, $\pi = 5$.

Several destroy and repair operators that we use are **Random removal (R_1)**: remove a randomly selected node from Sol_0, **Worst removal (R_2)**: remove a node with a high removal cost (the difference in objective function values between including and excluding a particular node), **Route removal (R_3)**: randomly select a vehicle and remove its visited nodes, **Greedy insertion (I_1)**: insert a node to a position with the lowest insertion cost (the difference in objective function values between after and before inserting a node to a particular position), **k-regret insertion (I_2, I_3, I_4)**: insert a node to a position with the

largest regret value (the difference in objective function values when a node is inserted in the best position and in the k-best position). We use $k = 2, 3$, and 4.

Each of the removed nodes is only considered as a candidate to be inserted in a route of Sol_0 if it satisfies both vehicle capacity and time horizon constraints. Therefore, the feasibility of Sol_0 is guaranteed, unless some of the removed nodes cannot be inserted to any positions in Sol_0. If that happens, a high penalty value is added to the objective function value (total cost TC). Sol_0 is accepted if and only if it improves Sol^*. Otherwise, Sol_0 is set to be Sol^*, such that a new neighborhood solution is always explored from Sol^*. Each of the operators' score s_j is then updated by following Eq. (1), where $\delta_1 > \delta_2$. We implemented 0.5 and 0 for δ_1 and δ_2 respectively.

$$s_j = \begin{cases} s_j + \delta_1, & \text{if } Sol_0 < Sol^* \\ s_j + \delta_2, & \text{if } Sol_0 \geq Sol^* \end{cases} \quad \forall j \in R \cup I \tag{1}$$

After η_{ALNS} iterations, each of the operators' weight w_j is updated by following Eq. (2), where γ refers to the reaction factor ($0 < \gamma < 1$) to control the influence of the recent success of an operator on its weight and χ_j is the frequency of using operator j. Consequently, each of the operators' probability p_j is updated by following Eq. (3). The ALNS is terminated when there is no solution improvement after $\eta \times \theta$ iterations. Upon this termination, the Sol^* constructed by ALNS becomes an upper bound of the VRPCD solution. It means that solving the following set partitioning formulation will only yield a lower (or at least the same) objective function value as the Sol^* constructed by ALNS.

$$w_j = \begin{cases} (1-\gamma)w_j + \gamma \frac{s_j}{\chi_j}, & \text{if } \chi_j > 0 \\ (1-\gamma)w_j, & \text{if } \chi_j = 0 \end{cases} \quad \forall j \in R \cup I \tag{2}$$

$$p_j = \begin{cases} \frac{w_j}{\sum_{k \in R} w_k} & \forall j \in R \\ \frac{w_j}{\sum_{k \in I} w_k} & \forall j \in I \end{cases} \tag{3}$$

Each candidate route r in Ω_s is associated to a transportation cost of $c_r^{'}$ and a transportation time of $t_r^{'}$, while each candidate route r in Ω_c is associated to a transportation cost of $c_r^{''}$ and a transportation time of $t_r^{''}$. Let $a_{ir}^{'}$ be a binary parameter equal to 1 if route r visits node i; 0 otherwise ($r \in \Omega_s, i \in S$) and $a_{ir}^{''}$ be a binary parameter equal to 1 if route r visits node i; 0 otherwise ($r \in \Omega_c, i \in C$). Several decision variables in the set partitioning formulation:

- $x_r^{'} = 1$ if route r is selected; 0 otherwise ($r \in \Omega_s$)
- $x_r^{''} = 1$ if route r is selected; 0 otherwise ($r \in \Omega_c$)
- Tp_{max} = the maximum transportation time for pickup process
- Td_{max} = the maximum transportation time for delivery process

The objective is to minimize the total of transportation and operational costs, as formulated in (4). All supplier and customer nodes must be visited, as required in (5) and (6) respectively. (7) limits the number of selected routes (i.e. does

not exceed the number of available vehicles). (8) and (9) records the maximum transportation time in pickup and delivery process respectively. Finally, the two processes must be done within the time horizon, as expressed in (10).

$$Min \quad \sum_{r\in\Omega_s} c_r^{'} x_r^{'} + \sum_{r\in\Omega_c} c_r^{''} x_r^{''} + H\left(\sum_{r\in\Omega_s} x_r^{'} + \sum_{r\in\Omega_c} x_r^{''}\right) \tag{4}$$

$$\sum_{r\in\Omega_s} a_{ir}^{'} x_r^{'} = 1 \quad \forall i \in S \tag{5}$$

$$\sum_{r\in\Omega_c} a_{ir}^{''} x_r^{''} = 1 \quad \forall i \in C \tag{6}$$

$$\sum_{r\in\Omega_s} x_r^{'} + \sum_{r\in\Omega_c} x_r^{''} \leq |V| \tag{7}$$

$$t_r^{'} x_r^{'} \leq Tp_{max} \quad \forall r \in \Omega_s \tag{8}$$

$$t_r^{''} x_r^{''} \leq Td_{max} \quad \forall r \in \Omega_c \tag{9}$$

$$Tp_{max} + Td_{max} \leq T_{max} \tag{10}$$

4 Computational Results

The matheuristic is tested on benchmark VRPCD instances with 10-nodes [5]. We report the average values found for all instances out of ten runs. Since the instances are small, we could use CPLEX and the mathematical model presented in [5] to obtain the optimal solution for all these instances. It should be noted, however, that these optimal solutions were not reported in the state of the art yet. In Table 1, we evaluate the performance of our approach and those of the state-of-the-art algorithms based on these optimal solutions. The matheuristic is implemented in C++ with CPLEX 12.9.0.0 to solve the set partitioning formulation. All experiments were performed on a computer with Intel Core i7-8700 CPU @ 3.20 GHz processor, 32.0 GB RAM. The parameter values are: γ: 0.7, θ: 20, η_{ALNS}: 200, η: $(|S| + |C|) \times 2$.

Our proposed matheuristic is able to obtain either the same or further improve the best known solutions (BKS) which are consolidated from the state-of-the-art algorithms. On average, we outperform the BKS with 1.5%. Moreover, we obtain the optimal solution for each instance. In terms of the average of CPU time, our proposed matheuristic spends 0.16 s while [5–7] use 2.02, 0.12 and 2.06 s respectively. The average calculation time for generating the optimal solutions with CPLEX takes 1.05 s.

Table 1. Total cost comparison of the matheuristic and state-of-the-art algorithms

Instance	[5]	[6]	[7]	BKS	Opt	Matheuristic	Gap BKS to Opt	Gap Matheuristic to Opt
1	7571.4	6847.6	6953.0	6847.6	6823.0	6823.0	0.4%	0.0%
2	7103.7	6816.8	6741.0	6741.0	6741.0	6741.0	0.0%	0.0%
3	9993.5	9615.6	9269.0	9269.0	9269.0	9269.0	0.0%	0.0%
4	8338.0	7289.7	7255.0	7255.0	7229.0	7229.0	0.4%	0.0%
5	8709.9	6599.0	6524.0	6524.0	6475.0	6475.0	0.8%	0.0%
6	9143.5	9324.6	7613.0	7613.0	7434.0	7434.0	2.4%	0.0%
7	12721.2	12083.0	11990.0	11990.0	11713.0	11713.0	2.4%	0.0%
8	9275.7	8719.6	8158.0	8158.0	8158.0	8158.0	0.0%	0.0%
9	8096.5	7362.2	7120.0	7120.0	6989.0	6989.0	1.9%	0.0%
10	7044.8	6204.5	6056.0	6056.0	5960.0	5960.0	1.6%	0.0%
11	8051.8	7635.3	7434.0	7434.0	6916.0	6916.0	7.5%	0.0%
12	8661.0	7867.2	7800.0	7800.0	7656.0	7656.0	1.9%	0.0%
13	7370.2	7097.9	6934.0	6934.0	6783.0	6783.0	2.2%	0.0%
14	7132.3	5208.0	4704.0	4704.0	4417.0	4417.0	6.5%	0.0%
15	7563.4	7103.2	7088.0	7088.0	7072.0	7072.0	0.2%	0.0%
16	9983.6	8768.7	8616.0	8616.0	8440.0	8440.0	2.1%	0.0%
17	9538.1	9003.0	9003.0	9003.0	9003.0	9003.0	0.0%	0.0%
18	8057.4	6887.5	6911.0	6887.5	6760.0	6760.0	1.9%	0.0%
19	9042.6	7123.0	7051.0	7051.0	7051.0	7051.0	0.0%	0.0%
20	10478.0	10471.0	10004.0	10004.0	9786.0	9786.0	2.2%	0.0%
21	8380.5	5431.4	4753.0	4753.0	4644.0	4646.0	2.3%	0.0%
22	9016.9	6908.0	6442.0	6442.0	6442.0	6442.0	0.0%	0.0%
23	9489.2	9224.1	9156.0	9156.0	9156.0	9156.0	0.0%	0.0%
24	12513.6	11976.0	11976.0	11976.0	11976.0	11976.0	0.0%	0.0%
25	7114.3	6638.0	6346.0	6346.0	6346.0	6346.0	0.0%	0.0%
26	8421.3	7216.9	6880.0	6880.0	6817.0	6817.0	0.9%	0.0%
27	10666.8	9709.8	9541.0	9541.0	9541.0	9541.0	0.0%	0.0%
28	10123.3	7408.0	7107.0	7107.0	6782.0	6782.0	4.8%	0.0%
29	7503.2	6748.5	6762.0	6748.5	6591.0	6591.0	2.4%	0.0%
30	7642.6	7304.4	6942.0	6942.0	6919.0	6919.0	0.3%	0.0%
Avg							1.5%	0.0%

5 Conclusion

We study the integration of vehicle routing problem with cross-docking (VRPCD). A matheuristic approach based on ALNS and set partitioning is proposed. Preliminary results show that the matheuristic outperforms the state-of-the-art algorithms in terms of both solution quality and computational time. Solving larger benchmark instances will be included in future work.

Acknowledgment. This research is supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant.

References

1. Apte, U.M., Viswanathan, S.: Effective cross docking for improving distribution efficiencies. Int. J. Logist. **3**(3), 291–302 (2000)

2. Archetti, C., Speranza, M.G.: A survey on matheuristics for routing problems. EURO J. Comput. Optim. **2**(4), 223–246 (2014). https://doi.org/10.1007/s13675-014-0030-7
3. Foster, B.A., Ryan, D.M.: An integer programming approach to the vehicle scheduling problem. J. Oper. Res. Soc. **27**(2), 367–384 (1976)
4. Grangier, P., Gendreau, M., Lehuédé, F., Rousseau, L.M.: The vehicle routing problem with cross-docking and resource constraints. J. Heuristics (2019). https://doi.org/10.1007/s10732-019-09423-y
5. Lee, Y.H., Jung, J.W., Lee, K.M.: Vehicle routing scheduling for cross-docking in the supply chain. Comput. Ind. Eng. **51**(2), 247–256 (2006)
6. Liao, C.J., Lin, Y., Shih, S.C.: Vehicle routing with cross-docking in the supply chain. Expert Syst. Appl. **37**(10), 6868–6873 (2010)
7. Yu, V.F., Jewpanya, P., Redi, A.A.N.P.: Open vehicle routing problem with cross-docking. Comput. Ind. Eng. **94**, 6–17 (2016)