Adaptive Large Neighborhood Search for Vehicle Routing Problem with Cross-Docking

Aldy Gunawan, Audrey Tedja Widjaja
School of Information Systems
Singapore Management University
Singapore, Singapore
aldygunawan,audreyw@smu.edu.sg

Pieter Vansteenwegen
Leuven Mobility Research Center, CIB
KU Leuven
Leuven, Belgium
pieter.vansteenwegen@kuleuven.be

Vincent F. Yu
Department of Industrial Management
NTUST
Taipei, Taiwan
vincent@mail.ntust.edu.tw

***Abstract*—Cross-docking is considered as a method to manage and control the inventory flow, which is essential in the context of supply chain management. This paper studies the integration of the vehicle routing problem with cross-docking, namely VRPCD which has been extensively studied due to its ability to reduce the overall costs occurring in a supply chain network. Given a fleet of homogeneous vehicles for delivering a single type of product from suppliers to customers through a cross-dock facility, the objective of VRPCD is to determine the number of vehicles used and the corresponding vehicle routes, such that the vehicle operational and transportation costs are minimized. An adaptive large neighborhood search (ALNS) algorithm is proposed to solve the available benchmark VRPCD instances. The experimental results show that ALNS is able to improve 80 (out of 90) best known solutions and obtain the same solution for the remaining 10 instances within short computational time. We also explicitly analyze the added value of using an adaptive scheme and the implementation of the acceptance criteria of Simulated Annealing (SA) into the ALNS, and also present the contribution of each ALNS operator towards the solution quality.**

***Index Terms*—cross-docking, vehicle routing problem, scheduling, adaptive large neighborhood search**

I. Introduction

A supply chain refers to a system that moves products or services from suppliers to customers. Traditionally, members of a supply chain only concern themselves and they do not consider the global optimization by factoring other members in the supply chain. Managing the flow of inventory, e.g. raw materials, work-in-process, and finish goods, is one of the major challenges in supply chain management [1].

A warehousing strategy, called cross-docking, has been widely adapted by many companies due to its ability to reduce inventory and travel time during the delivery process within a supply chain network. The supplied products from suppliers are consolidated inside the cross-dock facility before being sent to the customers. The consolidation inside the cross-dock facility should be done in a short time, usually less than 12 hours, or products are even dispatched immediately to the customers [2].

The integration of the vehicle routing problem with cross-docking (VRPCD) was first addressed by [3] to eliminate a long origin-to-destination distance. Nikolopoulou et al. [4] remarked the benefits of VRPCD compared to direct shipping for cases where supplier-customer pairs are located remotely and for the case of clustered node distributions. Wen et al. [5] studied the VRPCD by considering time windows (VRPCDTW).

In this paper, we focus on the VRPCD developed in [3]. The VRPCD network consists of a set of suppliers who deliver a single type of product to a set of customers through a single cross-docking facility. The products are consolidated inside the cross-dock before sending them to the customers. The objective is to minimize the total cost, consists of vehicle operational and transportation costs, within a given time horizon. A detailed problem definition is explained in Section II.

Lee et al. [3] proposed a tabu search (TS) algorithm to solve the VRPCD. Liao et al. [1] further developed a new TS algorithm (imp-TS) which reduces the number of vehicles used during the search process. By using the same benchmark VRPCD instances [3], this new TS algorithm is able to improve the solutions by 10-36% for different sizes of problems, with much less computational time. Yu et al. [6] designed a simulated annealing (SA) algorithm, which was originally developed to solve an open VRPCD (OVRPCD). Computational results on the benchmark VRPCD instances show that the proposed SA is able to obtain the same or better solutions for 78 out of 90 instances. Other papers focused on developing algorithms to solve the VRPCDTW, such as TS [5] and local search [7].

Shaw [8] firstly designed a large neighborhood search (LNS) algorithm to solve variants of the VRP, such as the capacitated VRP (CVRP) and the VRPTW. LNS explores a large neighborhood of the current solution by removing the visited customers in the routing plan iteratively, and reinserting them. Ropke and Pisinger [9] designed an adaptive LNS (ALNS) algorithm to solve the pickup and delivery problem with time windows (PDPTW). The algorithm incorporates several destroy and repair operators that are selected adaptively during the entire search process.

Other classes of VRPs have also been solved by ALNS, such as the pollution-routing problem (PRP) [10], the two-echelon VRP (2E-VRP) [11], the location routing problem (LRP) [11], and the VRP with drones [12]. Its outstanding results in solving various transportation and scheduling problems [13] motivated us to design an ALNS algorithm to solve the VRPCD.

We tested our proposed ALNS on the available benchmark VRPCD instances. Computational results show that our proposed ALNS outperforms all state-of-the-arts algorithms (TS [3], imp-TS [1], and SA [6]). Furthermore, we present the added value of using an adaptive scheme and the acceptance criteria of SA in the proposed algorithm. The contribution of each ALNS operator towards the solution quality is also discussed.

II. Problem Description

A VRPCD network is described as two directed graphs $G' = (S \cup \{0\}, A')$ and $G'' = (C \cup \{0\}, A'')$, where node 0 represents a cross-dock facility, $S = \{1, 2, \dots, |S|\}$ is a set of supplier nodes, and $C = \{1, 2, \dots, |C|\}$ is a set of customer nodes. $A' = \{(i, j) : i \neq j \in S \cup 0\}$ and $A'' = \{(i, j) : i \neq j \in C \cup 0\}$ refer to a set of arcs connecting two different nodes i and j. Each of the connected arcs include transportation costs $c^{'}_{ij}$ and $c^{''}_{ij}$, and transportation times $t^{'}_{ij}$ and $t^{''}_{ij}$ for connecting supplier or customer nodes, respectively. Fig. 1 illustrates a feasible solution.

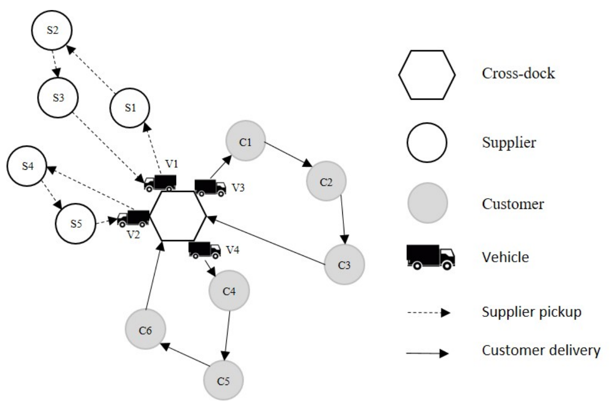


Fig. 1. Illustration of a VRPCD solution

A fleet of homogeneous vehicles $V = \{1, 2, \dots, |V|\}$ with capacity Q is available at the cross-dock facility. They are classified into pickup vehicles and delivery vehicles. The former departs from the cross-dock, visits one (or more) supplier(s), picks up a number of products P_i, and returns to the cross-dock. The products are then consolidated according to customers' demand inside the cross-dock facility. The latter departs from the cross-dock, visits one (or more) customer(s), delivers their demand D_i, and returns to the cross-dock. It is assumed that each vehicle only performs a single trip (e.g. departs from and return to the cross-dock at most once). A vehicle operational cost H is charged to each vehicle used. The consolidation process inside the cross-dock facility and the pickup and delivery processes at the nodes are assumed to be fast enough, and therefore it is not taken into consideration or already included in the transportation times.

The VRPCD aims to decide how many vehicles are assigned to perform the pickup and delivery processes, and also construct the corresponding routes, subject to the constraints as listed below:

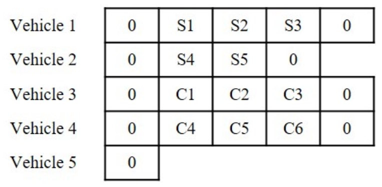
- the total transportation time for the pickup and delivery processes together does not exceed T_{max}. This implies that all products are first collected from the suppliers and then (after the consolidation) these products are distributed to the customers, all within T_{max}
- each supplier and customer can only be visited exactly once
- the number of utilized vehicles in pickup and delivery processes does not exceed $|V|$
- the amount of loads in each vehicle does not exceed Q

One may think that solving the VRPCD is similar to solving two independent VRPs. However, this is not the case due to the existence of the above-mentioned constraints about transportation time and number of vehicles which link these two VRPs together and definitely increase the complexity of the problem. The objective is to minimize both operational and transportation costs. For more details, readers are referred to [3] for the VRPCD mathematical model.

III. Proposed Algorithm

A. Solution Representation

The VRPCD solution is represented here as a two-dimensional matrix, where rows and columns indicate the vehicles and the route sequences, respectively. We set a fixed number of rows, which is the upper bound of the number of vehicles that can be used, $|V|$. However, each row may have a different number of columns which depends on the number of nodes visited by the corresponding vehicle. Fig. 2 illustrates a solution representation of the VRPCD network with $|S| = 5$, $|C| = 6$, and $|V| = 5$. It implies that vehicle 5 is not used and therefore, H is only charged for the four vehicles used. Vehicle 1 starts its route from the cross-dock (node 0), visits Suppliers 1, 2, and 3, and returns to the cross-dock. Vehicle 4 is used to visit Customers 4, 5 and 6. It is noted that the solution must satisfy (1), corresponding to the total transportation time constraint discussed in Section II.



Vehicle 1	0	S1	S2	S3	0
Vehicle 2	0	S4	S5	0	
Vehicle 3	0	C1	C2	C3	0
Vehicle 4	0	C4	C5	C6	0
Vehicle 5	0				

Fig. 2. Example of solution representation

$$\begin{aligned}
&\operatorname{argmax}\{(t^{'}_{0,S1} + t^{'}_{S1,S2} + t^{'}_{S2,S3} + t^{'}_{S3,0}),\\
&\qquad (t^{'}_{0,S4} + t^{'}_{S4,S5} + t^{'}_{S5,0})\} +\\
&\operatorname{argmax}\{(t^{''}_{0,C1} + t^{''}_{C1,C2} + t^{''}_{C2,C3} + t^{''}_{C3,0}),\\
&\qquad (t^{''}_{0,C4} + t^{''}_{C4,C5} + t^{''}_{C5,C6} + t^{''}_{C6,0})\} \leq T_{max}
\end{aligned} \tag{1}$$

The objective function value of the solution shown in Fig. 2, the total cost TC, is calculated by (2). It aims to minimize both operational cost (obtained by minimizing the number of vehicles used) and transportation cost (obtained by constructing the least arc cost routes).

$$\begin{aligned} TC =& 4 \times H + c'_{0,S1} + c'_{S1,S2} + c'_{S2,S3} + c'_{S3,0} + \\ & c'_{0,S4} + c'_{S4,S5} + c'_{S5,0} + \\ & c''_{0,C1} + c''_{C1,C2} + c''_{C2,C3} + c''_{C3,0} + \\ & c''_{0,C4} + c''_{C4,C5} + c''_{C5,C6} + c''_{C6,0} \end{aligned} \tag{2}$$

B. The Adaptive Large Neighborhood Search Algorithm

The ALNS pseudocode is presented in Algorithm 1.

Algorithm 1: ALNS pseudocode

```
1  $Sol_0, Sol^*, Sol'$ ← Initial Solution
2  $Temp \leftarrow T_0$
3  NoImpr, Iter ← 0
4  FoundBestSol ← False
5  Set $s_j$ and $w_j$ such that $p_j$ is equally likely
6  while NoImpr $< \theta$ do
7      RemovedNodes ← 0
8      while RemovedNodes $< \pi$ do
9          $Sol_0 \leftarrow$ Destroy($R_r$)
10         UpdateRemovedNodes(RemovedNodes, $R_r$)
11     end
12     while RemovedNodes $> 0$ do
13         $Sol_0 \leftarrow$ Repair($I_i$)
14         UpdateRemovedNodes(RemovedNodes, $I_i$)
15     end
16     AcceptanceCriteria($Sol_0, Sol^*, Sol', Temp$)
17     Update $s_j$
18     if Iter mod $\eta_{ALNS}$ = 0 then
19         Update $w_j$ and $p_j$
20     end
21     if Iter mod $\eta_{SA}$ = 0 then
22         if FoundBestSol = False then
23             NoImpr ← NoImpr + 1
24         end
25         else
26             NoImpr ← 0
27         end
28         FoundBestSol ← False
29         $Temp \leftarrow Temp \times \alpha$
30     end
31     Iter ← Iter + 1
32 end
33 Return $Sol^*$
```

The main idea of ALNS is to remove nodes from the solution by using DESTROY operators and reinsert them into a more profitable position by using REPAIR operators. A score is assigned to each operator in order to assess its performance upon generating a new neighborhood solution. The better the new generated solution is, the higher the score is given to the corresponding operator. After η_{ALNS} iterations, each operator's weight is calculated based on its score. The weight corresponds to the probability of choosing an operator over another. The better the performance is, the higher the chance to be selected in the next iteration.

Let $R = \{R_r | r = 1, 2, \ldots, |R|\}$ be a set of DESTROY operators and $I = \{I_i | i = 1, 2, \ldots, |I|\}$ be a set of REPAIR operators. All operators $j (j \in R \cup I)$ initially have the same weight w_j and probability p_j to be selected, such that (3) is satisfied.

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

The ALNS adopts the Simulated Annealing (SA) acceptance criteria, a worse solution may be accepted with a certain probability [14]. Therefore, each of the operator's score s_j is adjusted by (4).

$$s_j = \begin{cases} s_j + \delta_1, & \text{if the new solution is the} \\ & \text{best found solution so far} \\ s_j + \delta_2, & \text{if the new solution improves} \\ & \text{the current solution} \\ s_j + \delta_3, & \text{if the new solution does not} \\ & \text{improve the current solution,} \\ & \text{but it is accepted} \end{cases} \quad \forall j \in R \cup I \tag{4}$$

with $\delta_1 > \delta_2 > \delta_3$. We implemented 0.5, 0.33, and 0.17 for δ_1, δ_2, and δ_3 respectively. Then, the operator's weight w_j is adjusted by following (5).

$$w_j = \begin{cases} (1-\gamma) w_j + \gamma \frac{s_j}{\chi_j}, & \text{if } \chi_j > 0 \\ (1-\gamma) w_j, & \text{if } \chi_j = 0 \end{cases} \quad \forall j \in R \cup I \tag{5}$$

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.

Let Sol_0, Sol^*, and Sol' be the current solution, the best found solution so far, and the starting solution at each iteration respectively. At the beginning, Sol_0, Sol^*, and Sol' are set to be the same as the initial solution. The initial solution is constructed by assigning nodes to a vehicle without violating the vehicle capacity. Then, the route sequence of each vehicle is constructed by inserting every time the next least transportation cost node. When T_{max} is violated, we reduce the trip of a vehicle that consumes the highest transportation time by transferring some of its visited nodes to other vehicles. The current temperature $(Temp)$ is set to be equal to the initial temperature (T_0) and will be decreased by α after η_{SA} iterations. The score s_j and weight w_j of each operator j are set such that the probability of choosing an operator p_j is equally likely in the beginning.

At each iteration, the Sol_0 is modified by applying DESTROY and REPAIR operators. π nodes are removed from Sol_0 by using a selected DESTROY operator R_i. R_i is applied until π is reached. Then, the removed π nodes are reinserted to Sol_0 by a selected REPAIR operator I_i. We use $\pi = 5$. 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 the removed nodes cannot be inserted to any positions in Sol_0. If that happens, a high penalty value (e.g. a very large number) is added to the objective function value. The details of ALNS operators are introduced in Section III-C.

Sol_0 is directly accepted if its objective function value is better than Sol^* or Sol'. Otherwise, it will only be accepted with probability $e^{\frac{-(Sol_0 - Sol')}{Temp}}$. Each of the operator's score s_j is then updated according to (4). After η_{ALNS} iterations, each of the operator's weight w_j is updated by (5) and its probability p_j is updated according to (3). The ALNS is terminated when there is no solution improvement after θ successive temperature reductions.

C. The Adaptive Large Neighborhood Search Operators

Random removal (R_1): select one node randomly and remove from the current solution. Once R_1 is applied, REMOVEDNODES is increased by 1.

Worst removal (R_2): remove the node with the x^{th} highest removal cost. The removal cost is defined as the difference in objective function values between including and excluding a particular node. For each nodes in Sol_0, the removal cost is calculated and sorted in a descending order. x is determined by (6), in which the value ranges between 1 to ξ, with $x = 1$ having the highest chance to be drawn, and the chance is decreasing for larger xs. It means that R_2 tries to remove a node which contributes a high cost if it is located in the current position. Instead of always choosing a node with the highest removal cost (e.g. $x = 1$), R_2 also considers other nodes in order to diversify the removal process (in which $2 \leq x \leq \xi$), but with a lower chance. We implement $p = 3$. y_1 is introduced to maintain the randomness, $y_1 \sim U(0, 1)$. ξ is the number of candidate nodes which is formally formulated in (7) case 1. By referring to Algorithm 1 lines 8 – 11, if we consider $|C| = 6$, $|S| = 4$ with REMOVEDNODES = 0, the number of candidate nodes that can be removed in the current iteration is $|C| + |S| - 0 = 10$. Once a node is removed, REMOVEDNODES is increased by 1, then in the next iteration, the number of candidate nodes that can be removed becomes $|C| + |S| - 1 = 9$. It means that every time a node is removed from the Sol_0, the number of candidate nodes that can be removed from the Sol_0 in the next iteration is decreased.

$$x = \lceil y_1^p \times \xi \rceil \tag{6}$$

$$\xi = \begin{cases} |C| + |S| - \text{REMOVEDNODES}, & \text{for } R_2 \\ |C| + |S| - \text{REMOVEDNODES} - 2, & \text{for } R_4, R_5 \\ \text{REMOVEDNODES}, & \text{for } I_9 \end{cases} \tag{7}$$

Route removal (R_3): randomly select a vehicle and remove its z visited nodes. z is determined by (8), where β is the number of nodes visited by the corresponding vehicle. Therefore, once R_3 is applied, REMOVEDNODES is increased by z. R_3 is implemented to speed up the removal process compared to the one of R_1. When implementing R_1 requires at least π iterations (see Algorithm 1 lines 8 – 11), the implementation of R_3 may only require one iteration if $\beta \geq \pi$ (see (8) case 1). The worst case is that if all vehicles only visit one node each, then, the implementation of R_3 would also need π iterations (see (8) case 2).

$$z = \begin{cases} \pi, & \text{if } \beta \geq \pi \\ \beta, & \text{otherwise} \end{cases} \tag{8}$$

Historical node pair removal (R_4): remove a pair of nodes with the x^{th} highest transportation cost. x is determined by (6) while ξ is determined by (7) case 2. However, it should be noted that applying R_4 means that two nodes are removed simultaneously from the Sol_0 in one iteration and therefore, REMOVEDNODES is increased by 2. When applying R_4 is impossible (e.g. when REMOVEDNODES = 4 while $\pi = 5$), then, we apply R_1 instead. R_4 tries to remove two adjacent nodes with a high transportation cost from the Sol_0, such that when REPAIR reinserts them back to Sol_0, they can be located in better, and probably separated, positions.

Worst pair removal (R_5): similar to R_2, but instead of only one node, R_5 chooses a pair of nodes. The underlying difference between R_4 and R_5 is that R_4 only focuses in the transportation cost between two nodes, while R_5 considers the overall costs. Same as R_4, applying R_5 means that two nodes are removed simultaneously from the Sol_0 in one iteration and therefore, REMOVEDNODES is increased by 2. When applying R_5 is impossible, we apply R_1 instead. x is determined by (6) while ξ is determined by (7) case 2.

Shaw removal (R_6): remove a node that is highly related with other removed nodes in a predefined way. R_6 tries to remove some similar nodes, such that it is easier to replace the positions of one another during the repair process. The last removed node is denoted as node i while the next candidate of the removed node is denoted as node j. The relatedness value (φ_j) of node j to node i is calculated by (9). R_6 starts by randomly selecting a node to be removed, set this node as i, and then calculating φ_j for the remaining nodes in Sol_0. The next removed node is the node with the lowest φ_j. Since R_6 removes one node at a time, REMOVEDNODES is increased by 1. The ϕ_1 to ϕ_4 are weights given to each of the relatedness components, in which we set as 0.25 each. $l_{ij} = -1$ if nodes i and j are in the same vehicle; 1 otherwise.

$$\varphi_j = \begin{cases} \phi_1 c'_{ij} + \phi_2 t'_{ij} + \phi_3 l_{ij} + \phi_4 |P_i - P_j|, & \text{if } i \in S \\ \phi_1 c''_{ij} + \phi_2 t''_{ij} + \phi_3 l_{ij} + \phi_4 |D_i - D_j|, & \text{if } i \in C \end{cases} \tag{9}$$

Greedy insertion (I_1): insert a node to a position resulting in the lowest insertion cost (i.e. the difference in objective function value after and before inserting a node to a particular position). For each of the removed nodes, the insertion cost to all possible positions is calculated and sorted in an ascending order. The node with the lowest insertion cost is then selected and inserted to the corresponding position. Once I_1 inserts one node to the Sol_0, REMOVEDNODES is decreased by 1.

***k*-regret insertion** (I_2, I_3, I_4): the regret value is calculated by the difference in Total Cost (TC) when node j is inserted in the best position (denoted as $TC_1(j)$) and in the k-best position (denoted as $TC_k(j)$). The idea is to select a node which leads to the largest regret value if it is not inserted in its best position, which is formally formulated in (10). This node is then inserted in its best position. In other words, this operator tries to insert the node that we will regret the most if it is not inserted now [9]. We implemented $k = 2, 3$ and 4. Once I_2, I_3, or I_4 inserts one node to the Sol_0, RemovedNodes is decreased by 1.

$$\underset{j \in \text{RemovedNodes}}{\operatorname{argmax}} \left\{ \sum_{i=2}^{k} (TC_i(j) - TC_1(j)) \right\} \qquad (10)$$

Greedy insertion with noise function (**I_5**): an extension of I_1. A noise function is applied to the objective function value (11) when selecting the best position of a node [10], where $\overline{\mathrm{e}}$ is the maximum transportation cost between nodes (problem-dependent), μ is a noise parameter that we set to 0.1, and $y_2 \sim U(-1, 1)$.

$$TC_{new} = TC + \overline{\mathrm{e}} \times \mu \times y_2 \qquad (11)$$

k-regret insertion with noise function (**I_6, I_7, I_8**): an extension of I_2, I_3, and I_4 by applying a noise function to the objective function value (11) when calculating the regret value [10].

GRASP insertion (**I_9**): similar to I_1, but instead of choosing a node with the lowest insertion cost, I_9 chooses the node with the x^{th} lowest insertion cost. x is determined by (6) while ξ is determined by (7) case 3.

IV. Computational Results

A. Benchmark Instances

Benchmark VRPCD instances are available online in http://web.ntust.edu.tw/~vincent/ovrpcd/. They are grouped by the number of nodes ($|S| + |C|$): 10-nodes, 30-nodes, and 50-nodes. The parameter values are listed in Table I.

TABLE I
VRPCD parameter values [3]

Parameter	Set 1	Set 2	Set 3
$\|S\|$	4	7	12
$\|C\|$	6	23	38
$\|V\|$	10	20	30
Q	70	150	150
H	1000	1000	1000
T_{max}	960	960	960
$e_{ij}^{'}$, $e_{ij}^{''}$	U∼(48,560)	U∼(48,480)	U∼(48,560)
$t_{ij}^{'}$, $t_{ij}^{''}$	U∼(20,200)	U∼(20,100)	U∼(20,200)
P_i, D_i	U∼(5,50)	U∼(5,20)	U∼(5,30)

B. Results and Discussion

The proposed ALNS is coded in C++. The experiments were performed on a computer with Intel Core i7-8700 CPU @ 3.20 GHz processor, 32.0 GB RAM. The ALNS parameter values are $\gamma = 0.7, \theta = 20, T_0 = 500, \alpha = 0.9, \eta_{ALNS} = 200, \eta_{SA} = (|S| + |C|) \times 2$. We perform 10 replications for each instance and record the average results.We compare our results against those of state-of-the-art algorithms: tabu search (TS) [3], improved tabu search (imp-TS) [1] and simulated annealing (SA) [6], in terms of the solution quality and CPU time (in seconds), as summarized in Table II and III respectively. We converted the CPU times of the state-of-the-art algorithms, based on https://cpu.userbenchmark.com/, to make a fair comparison. The Improvement (%) made by ALNS towards the best known solution (BKS) is calculated by (12). ALNS is able to either improve the BKS (for 80 instances) or obtain the same solution as the BKS (for the remaining 10 instances). On average, ALNS improves the BKS up to 1.4%, 13.6%, and 21.8% for three sets of instances respectively.

$$Improvement(\%) = \frac{(TC_{ALNS} - TC_{BKS})}{TC_{BKS}} \times 100 \qquad (12)$$

Compared against TS [3], ALNS is able to obtain better total cost for all instances. The improvement is 32.87% on average, with slightly longer computational time. Compared against the imp-TS [1], ALNS performs better in solving 88 instances while obtains the same solution for another two instances, with an improvement of 15.72%, although with longer computational time. Compared to SA [6], ALNS improves the solution of 80 instances and obtains the same solution for the remaining 10 instances, with an improvement of 12.54% and faster computational time. We conclude that our proposed ALNS outperforms the state-of-the-art algorithms.

C. The Importance of the Adaptive Scheme

Here, we assess the importance of applying the adaptive scheme during the search process. We set the static probability (13) and (14) as the baseline, in which during the search process, each operator is equally likely to be selected. In order to assess adaptive performance, we calculate the gap between TC obtained by the adaptive scheme (TC_{ALNS}) towards the TC obtained by the baseline (static) scheme ($TC_{baseline}$), using (15).

$$p(R_i) = \frac{1}{|R|} \;\; \forall i \in R \qquad (13)$$

$$p(I_i) = \frac{1}{|I|} \;\; \forall i \in I \qquad (14)$$

$$Gap(\%) = \frac{(TC_{ALNS} - TC_{baseline})}{TC_{baseline}} \times 100 \qquad (15)$$

In most instances (see Fig. 3), the gap obtained is negative, which implies that the adaptive scheme outperforms the static scheme. However, due to the randomness aspect in selecting the operators, the adaptive scheme does not always outperform the static scheme. Also worse performing operators can sometimes be selected. The adaptive scheme is worth to use as it improves the baseline in all sets on average up to 0.51%, 0.67%, and 0.41% for Set 1, Set 2, and Set 3 respectively, without significant difference in computational time.

TABLE II
TOTAL COST COMPARISON OF PROPOSED ALNS AND STATE-THE-ART ALGORITHMS

Instance	**[3]**			**[1]**			**[6]**			**BKS**			**ALNS**			**Improvement (%)**		
	Set 1	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**
1	7571.4	12366.7	24284.6	6847.6	7692.9	20704.6	6953.0	7550.2	19804.5	6847.6	7550.2	19804.5	**6823.0**	**6678.5**	**16006.4**	**-0.4**	**-11.5**	**-19.2**
2	7103.7	14173.0	23435.6	6816.8	7787.2	20816.8	6741.0	7832.7	18248.1	6741.0	7787.2	18248.1	6741.0	**6702.8**	**14243.4**	0.0	**-13.9**	**-21.9**
3	9993.5	13836.8	23449.4	9615.6	7893.6	19612.2	9269.0	7747.4	18133.9	9269.0	7747.4	18133.9	9269.0	**6669.6**	**13969.9**	0.0	**-13.9**	**-23.0**
4	8338.0	10995.4	23471.1	7289.7	7792.2	19549.0	7255.0	7677.4	19083.6	7255.0	7677.4	19083.6	**7229.0**	**6379.3**	**15190.5**	**-0.4**	**-16.9**	**-20.4**
5	8709.9	11757.8	23406.2	6599.0	7224.8	20448.0	6524.0	7579.7	18877.3	6524.0	7224.8	18877.3	**6475.0**	**6296.7**	**14801.5**	**-0.8**	**-12.8**	**-21.6**
6	9143.5	11027.7	24026.6	9324.6	7245.9	21212.0	7613.0	7053.0	19783.0	7613.0	7053.0	19783.0	**7434.0**	**5656.7**	**15383.8**	**-2.4**	**-19.8**	**-22.2**
7	12721.2	11899.2	24190.0	12083.0	8206.9	20640.2	11990.0	7720.1	19690.0	11990.0	7720.1	19690.0	**11713.0**	**6938.0**	**16035.2**	**-2.3**	**-10.1**	**-18.6**
8	9275.7	12825.5	23158.9	8719.6	7880.9	20664.1	8158.0	7709.8	18939.2	8158.0	7709.8	18939.2	8158.0	**6733.7**	**14738.6**	0.0	**-12.7**	**-22.2**
9	8096.5	12718.6	23594.7	7362.2	8157.3	18920.0	7120.0	7882.5	18510.6	7120.0	7882.5	18510.6	**6989.0**	**6853.1**	**14258.5**	**-1.8**	**-13.1**	**-23.0**
10	7044.8	11794.7	23530.5	6204.5	7924.7	20384.2	6056.0	7734.9	19607.3	6056.0	7734.9	19607.3	**5960.0**	**6787.0**	**15903.0**	**-1.6**	**-12.3**	**-18.9**
11	8051.8	12094.9	23371.7	7635.3	7452.6	19941.6	7434.0	7721.7	18675.8	7434.0	7452.6	18675.8	**6916.0**	**6425.1**	**14922.6**	**-7.0**	**-13.8**	**-20.1**
12	8661.0	12132.5	21082.8	7867.2	8320.0	17258.4	7800.0	7899.8	17550.3	7800.0	7899.8	17258.4	**7656.0**	**6725.2**	**13318.5**	**-1.8**	**-14.9**	**-22.8**
13	7370.2	13223.4	21610.7	7097.9	8222.7	17829.9	6934.0	7863.7	18039.2	6934.0	7863.7	17829.9	**6783.0**	**6770.5**	**13435.3**	**-2.2**	**-13.9**	**-24.6**
14	7132.3	12413.9	23397.9	5208.0	8211.7	19845.2	4704.0	8141.1	18252.5	4704.0	8141.1	18252.5	**4417.0**	**6776.3**	**13818.7**	**-6.1**	**-16.8**	**-24.3**
15	7563.4	12521.4	24041.9	7103.2	8144.6	21863.0	7088.0	7941.6	19803.8	7088.0	7941.6	19803.8	**7072.0**	**6712.3**	**15504.6**	**-0.2**	**-15.5**	**-21.7**
16	9983.6	12044.4	22893.4	8768.7	7451.7	20144.2	8616.0	7901.9	18808.3	8616.0	7451.7	18808.3	**8440.0**	**6788.3**	**14427.6**	**-2.0**	**-8.9**	**-23.3**
17	9538.1	12699.4	22950.4	9003.0	8086.2	20093.3	9003.0	8055.0	18713.8	9003.0	8055.0	18713.8	9003.0	**6727.4**	**14692.6**	0.0	**-16.5**	**-21.5**
18	8057.4	11001.4	24358.2	6887.5	7576.0	20244.8	6911.0	7798.3	18579.7	6887.5	7576.0	18579.7	**6760.0**	**6682.1**	**14410.1**	**-1.9**	**-11.8**	**-22.4**
19	9042.6	12724.4	25068.7	7123.0	7871.2	19955.0	7051.0	7964.3	18453.2	7051.0	7871.2	18453.2	7051.0	**6826.9**	**15111.9**	0.0	**-13.3**	**-18.1**
20	10478.0	12357.7	23232.1	10471.0	7883.7	19267.7	10004.0	7522.4	18167.3	10004.0	7522.4	18167.3	**9786.0**	**6820.5**	**14066.8**	**-2.2**	**-9.3**	**-22.6**
21	8380.5	13177.0	22564.8	5431.4	7914.1	19533.4	4753.0	7886.2	19226.0	4753.0	7886.2	19226.0	**4644.2**	**6679.5**	**14299.2**	**-2.3**	**-15.3**	**-25.6**
22	9016.9	11545.0	24360.7	6908.0	8005.3	19032.1	6442.0	7841.1	18551.6	6442.0	7841.1	18551.6	6442.0	**6716.1**	**14330.8**	0.0	**-14.3**	**-22.8**
23	9489.2	12308.1	24377.9	9224.1	7883.5	20562.5	9156.0	7791.5	18514.8	9156.0	7791.5	18514.8	9156.0	**6600.3**	**14642.0**	0.0	**-15.3**	**-20.9**
24	12513.6	12722.7	22008.7	11976.0	7731.2	19288.2	11976.0	7957.8	18558.5	11976.0	7731.2	18558.5	11976.0	**6615.1**	**13690.2**	0.0	**-14.4**	**-26.2**
25	7114.3	12844.9	24256.6	6638.0	7884.8	19695.9	6346.0	7839.4	18574.2	6346.0	7839.4	18574.2	6346.0	**6799.8**	**14716.2**	0.0	**-13.3**	**-20.8**
26	8421.3	13297.5	23424.9	7216.9	8001.6	20610.5	6880.0	7846.2	18995.7	6880.0	7846.2	18995.7	**6817.0**	**6971.5**	**14803.2**	**-0.9**	**-11.1**	**-22.1**
27	10666.8	13415.2	22961.4	9709.8	8899.4	18942.8	9541.0	8128.6	18128.7	9541.0	8128.6	18128.7	9541.0	**7215.7**	**14711.4**	0.0	**-11.2**	**-18.9**
28	10123.3	12613.0	23822.3	7408.0	10131.0	20097.3	7107.0	8367.5	18952.4	7107.0	8367.5	18952.4	**6782.0**	**7279.0**	**14546.7**	**-4.6**	**-13.0**	**-23.2**
29	7503.2	12840.8	23678.3	6748.5	8276.9	22248.1	6762.0	8003.1	19056.1	6748.5	8003.1	19056.1	**6591.0**	**6917.8**	**14856.4**	**-2.3**	**-13.6**	**-22.0**
30	7642.6	13796.2	23149.8	7304.4	8251.6	19321.9	6942.0	7760.9	18268.6	6942.0	7760.9	18268.6	**6919.0**	**6648.5**	**14654.2**	**-0.3**	**-14.3**	**-19.8**
Avg																**-1.4**	**-13.6**	**-21.8**

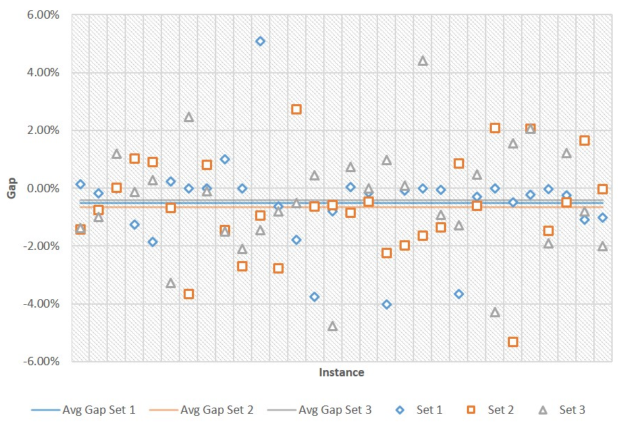


Fig. 3. Gap comparison with and without adaptive scheme (a negative gap indicates the adaptive scheme performs better)

D. The Importance of the SA Acceptance Criteria

Here, we assess the importance of embedding the SA acceptance criteria into the ALNS, which allows ALNS to accept a worse solution within a certain probability. We define the baseline approach as when the ALNS only accepts a better solution. For comparison purpose, we record the TC obtained by Algorithm 1 (TC_{ALNS}), and also the TC obtained by the baseline approach ($TC_{baseline}$). We again calculate the gap between TC_{ALNS} and $TC_{baseline}$ by (15).

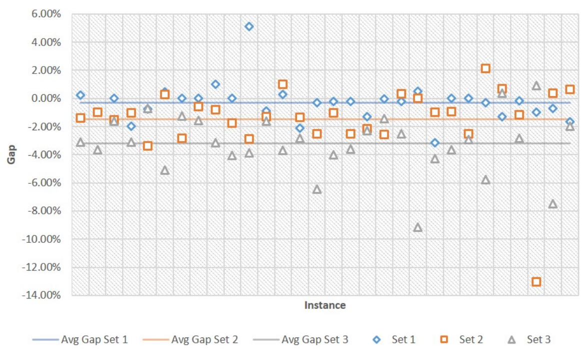


Fig. 4. Gap comparison with and without SA acceptance criteria (a negative gap indicates using SA acceptance criteria performs better)

The SA acceptance criteria plays an important role towards the TC obtained, as can be observed from Fig. 4 since most of the gap values are negative. The SA acceptance criteria significantly affects TC especially for larger instances. This is due to its ability to avoid a local optima solution such that it may search a vast search space, rather than always focus on one space. The improvement made by the SA acceptance criteria is up to 0.32%, 1.46%, and 3.21% for Set 1, Set 2, and Set 3, respectively. No significant difference in computational time is observed for both scenarios.

E. Analysis of ALNS Operators

Additional experiments are conducted to evaluate the importance of each ALNS operator introduced in Section III-C. We calculate the gap difference (15) between using all operators (TC_{ALNS}) and removing one of the operators ($TC_{baseline}$). In total, there are 15 baseline combinations, as listed in Table IV. We randomly select a subset of instances, ranged from Set 1 to Set 3, and the average gap of each combination is illustrated in Fig. 5.

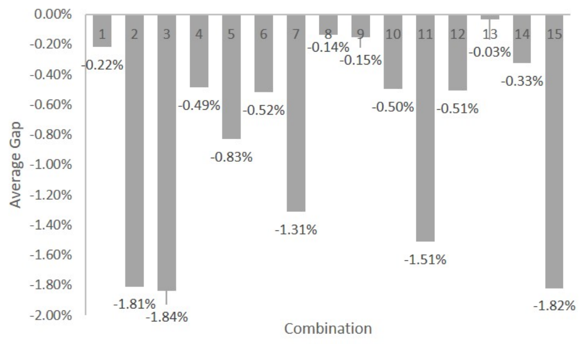


Fig. 5. Average gap of each ALNS operator combination

All different combinations produce negative average gap values. By removing one particular operator, the performance of ALNS is deteriorated. Operators R_2, R_3, I_1, I_5, I_9 play an important role to improve the TC. Even though some operators (such as I_2, I_3, I_7) are not significantly improve the TC, they still reduce the TC, and therefore worth to be implemented. Removing one of the ALNS operator does not significantly affect the computational time.

V. Conclusion

We study an integration of vehicle routing problem with cross-docking (VRPCD). Given a fleet of homogeneous vehicles, products supplied by the suppliers are delivered trough a cross-dock facility before sending them to the customers. The objective is to determine the number of vehicles used to perform the two processes and the corresponding vehicle routes, such that the operational and transportation costs are minimized, respecting the time horizon and vehicle capacity. An ALNS algorithm which employs different DESTROY operators and REPAIR operators is designed. The idea is to remove some nodes from the solution repetitively and to insert them to a more profitable position. The SA framework is embedded to discover a vast search space during the search process.

Assessing the performance of the proposed ALNS through the benchmark VRPCD instances, it improves the BKS for 80 out of 90 instances and obtains the same solution for the remaining 10 instances. However, the parameter value selection could be improved. It can be done by splitting the instances into training and testing sets to avoid overfitting and to determine the collaboration aspect between operators [15]. It would be interesting to group customers/suppliers first since they may be too far, then the route sequence can focus on a particular cluster. Generating and solving larger instances, e.g. with 100 or 200 nodes, would be interesting for future research as well.

TABLE III
CPU TIME COMPARISON OF PROPOSED ALNS AND STATE-OF-THE-ART ALGORITHMS

Instance	**[3]**			**[1]**			**[6]**			**ALNS**		
	Set 1	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**
1	1.52	3.00	5.62	0.22	0.37	0.49	1.91	2.80	7.49	0.22	1.03	3.63
2	1.74	3.55	5.73	0.23	0.16	0.64	1.78	2.90	6.62	0.15	1.04	3.60
3	2.37	4.32	5.80	0.19	0.43	0.30	2.44	3.00	7.10	0.14	1.05	3.57
4	1.60	2.09	5.87	0.27	0.23	0.44	1.75	3.00	6.89	0.14	0.86	3.37
5	2.28	2.26	7.28	0.21	0.39	0.61	1.75	3.50	7.09	0.15	1.29	4.57
6	1.82	2.10	5.38	0.03	0.22	0.56	1.81	3.10	7.47	0.11	0.86	3.95
7	2.80	2.61	7.65	0.01	0.11	0.55	2.77	3.00	6.11	0.13	0.99	4.29
8	1.85	3.00	9.88	0.04	0.16	0.42	2.21	3.30	6.91	0.14	1.00	3.15
9	2.04	3.10	5.54	0.25	0.16	0.38	1.75	2.80	7.11	0.12	0.84	3.89
10	1.82	2.38	5.77	0.36	0.20	0.52	2.05	2.60	7.10	0.12	1.01	4.01
11	1.80	3.03	5.37	0.00	0.29	0.37	1.93	3.00	7.15	0.12	0.80	3.56
12	1.72	2.64	4.46	0.24	0.16	0.17	1.76	2.90	6.75	0.13	0.93	3.85
13	1.54	2.92	4.62	0.24	0.15	0.16	1.88	2.90	6.32	0.13	0.96	3.82
14	1.53	2.76	5.44	0.00	0.18	0.53	1.69	3.60	7.38	0.11	0.94	3.60
15	1.61	3.06	6.31	0.41	0.38	0.55	1.75	3.10	7.97	0.12	0.99	4.32
16	2.05	3.15	5.18	0.03	0.28	0.30	2.05	3.10	6.95	0.11	0.81	3.81
17	2.26	2.14	6.93	0.06	0.34	0.44	2.60	3.00	6.72	0.13	0.89	3.86
18	1.74	1.70	6.25	0.19	0.28	0.53	1.88	2.90	7.07	0.13	0.83	4.00
19	2.21	2.77	5.68	0.03	0.36	0.28	1.92	3.10	7.93	0.11	1.06	3.72
20	2.55	2.72	4.79	0.00	0.24	0.36	2.32	2.50	7.56	0.11	0.91	3.95
21	2.06	3.39	5.43	0.00	0.26	0.61	1.51	3.40	6.97	0.12	0.97	4.76
22	2.42	2.43	6.04	0.01	0.35	0.37	1.85	2.90	7.83	0.12	0.95	3.90
23	2.31	2.93	5.88	0.11	0.38	0.51	2.28	3.20	7.35	0.14	0.96	4.04
24	2.64	2.87	5.36	0.00	0.48	0.05	2.78	3.00	6.81	0.11	1.01	3.80
25	1.68	2.67	5.76	0.10	0.20	0.33	1.94	3.00	7.55	0.13	0.75	4.15
26	2.04	3.31	5.05	0.03	0.16	0.14	1.86	3.30	7.80	0.13	1.08	3.65
27	2.47	3.25	5.17	0.06	0.17	0.31	2.39	3.30	7.12	0.13	0.78	3.70
28	2.69	2.60	5.56	0.01	0.00	0.39	1.84	2.80	7.41	0.11	1.14	3.49
29	1.73	3.28	64.84	0.18	0.38	0.20	1.86	2.90	6.96	0.14	0.90	3.54
30	1.82	3.78	5.91	0.09	0.28	0.65	3.51	2.70	8.29	0.13	0.87	3.17
Avg	0.51	0.72	1.96	0.03	0.07	0.10	1.37	2.01	4.79	0.13	0.95	3.82

TABLE IV
ALNS OPERATOR COMBINATIONS

Combination	**Operators employed**
1	$\{R_2, R_3, R_4, R_5, R_6, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$
2	$\{R_1, R_3, R_4, R_5, R_6, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$
3	$\{R_1, R_2, R_4, R_5, R_6, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$
4	$\{R_1, R_2, R_3, R_5, R_6, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$
5	$\{R_1, R_2, R_3, R_4, R_6, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$
6	$\{R_1, R_2, R_3, R_4, R_5, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$
7	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$
8	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_1, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$
9	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_1, I_2, I_4, I_5, I_6, I_7, I_8, I_9\}$
10	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_1, I_2, I_3, I_5, I_6, I_7, I_8, I_9\}$
11	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_1, I_2, I_3, I_4, I_6, I_7, I_8, I_9\}$
12	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_1, I_2, I_3, I_4, I_5, I_7, I_8, I_9\}$
13	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_1, I_2, I_3, I_4, I_5, I_6, I_8, I_9\}$
14	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_9\}$
15	$\{R_1, R_2, R_3, R_4, R_5, R_6, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8\}$

ACKNOWLEDGMENT

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

REFERENCES

[1] C. J. Liao, Y. Lin, and S. C. Shih, "Vehicle routing with cross-docking in the supply chain," *Expert Systems with Applications*, vol. 37, no. 10, pp. 6868–6873, 2010.

[2] U. M. Apte and S. Viswanathan, "Effective cross docking for improving distribution efficiencies," *International Journal of Logistics*, vol. 3, pp. 291–302, 2000.

[3] Y. H. Lee, J. W. Jung, and K. M. Lee, "Vehicle routing scheduling for cross-docking in the supply chain," *Computers and Industrial Engineering*, vol. 51, no. 2, pp. 247–256, 2006.

[4] A. I. Nikolopoulou, P. P. Repoussis, C. D. Tarantilis, and E. E. Zachariadis, "Moving products between location pairs: Cross-docking versus direct-shipping," *European Journal of Operational Research*, vol. 256, no. 3, pp. 803–819, 2017.

[5] M. Wen, J. Larsen, J. Clausen, J. F. Cordeau, and G. Laporte, *Journal of the Operational Research Society*, vol. 60, no. 12, pp. 1708–1718, 2009.

[6] V. F. Yu, P. Jewpanya, and A. A. N. P. Redi, "Open vehicle routing problem with cross-docking," *Computers and Industrial Engineering*, vol. 94, pp. 6–17, 2016.

[7] J. M. Urtasun and E. Montero, "An study of operator design under an adaptive approach for solving the cross-docks vehicle routing problem," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 2098–2105.

[8] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *International conference on principles and practice of constraint programming*. Springer, 1998, pp. 417–431.

[9] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation science*, vol. 40, no. 4, pp. 455–472, 2006.

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

[11] V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic, "An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics," *Computers & operations research*, vol. 39, no. 12, pp. 3215–3228, 2012.

[12] D. Sacramento, D. Pisinger, and S. Ropke, "An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 289–315, 2019.

[13] D. Pisinger and S. Ropke, "Large neighborhood search," in *Handbook of metaheuristics*. Springer, 2010, pp. 399–419.

[14] R. Lutz, "Adaptive large neighborhood search," 2015.

[15] C.-Y. Chuang and S. Smith, "Learning and utilizing interaction patterns among neighborhood-based heuristics," in *Twelfth Annual Symposium on Combinatorial Search*, 2019.