RAIRO-Oper. Res. 55 (2021) S2291–S2323
https://doi.org/10.1051/ro/2020087

RAIRO Operations Research
www.rairo-ro.org

VARIABLE NEIGHBORHOOD SEARCH BASED ALGORITHMS FOR CROSSDOCK TRUCK ASSIGNMENT

Cécilia Daquin*, Hamid Allaoui, Gilles Goncalves and Tienté Hsu

Abstract. To operate a cross-dock successfully, an efficient assignment of trucks to docks is one of the key decisions. In this paper, we are interested in the cross-dock assignment of trucks to docks problem, where the number of trucks exceeds the number of docks. The objective is to minimize the cost of transferring goods within the cross-dock while avoiding delivery penalties. This problem being NP-hard, we use Variable Neighborhood Search metaheurisitc (VNS) to solve it approximately. More specifically, we conduct a structured empirical study to compare several VNS configurations and to find which is/are the most effective for this cross-dock problem. In this work, first we analyze the way the search strategy and the neighborhood operators can be combined in a VNS framework according to their efficiency within a local search. Then the best configurations are tested within three VNS variants, namely Basic VNS (BVNS), General VNS (GVNS) using Basic VND (B-VND) and GVNS using Union VND (U-VND) according to the number of used operators and the order of applying these operators. Finally we evaluate the influence of the stopping criterion within these variants. Some significant differences among these configurations are shown and illustrated by conducting the Friedman test.

Mathematics Subject Classification. 90B06.

Received September 23, 2019. Accepted August 3, 2020.

1. Introduction

Cross-docking is a common logistics practice in industry where goods are received from suppliers in inbound trucks and loaded directly without storage into outbound trucks to customers. It enables to reduce operational costs while shortening delivery time and increasing goods flow (Fig. 1). A cross-dock includes various docks to which trucks can be assigned for unloading and loading operations. Inbound trucks are assigned to docks to unload received goods from suppliers. Then goods are immediately transferred to appropriate docks where outbound trucks are assigned. Finally, goods are loaded for delivering customers. Once an inbound truck is completely unloaded or an outbound truck is completely loaded, the dock is available for another truck. The problem is how to assign docks to trucks while synchronizing the arrival and the departure of cargos This property allows cost and delivery lead-time to be reduced. Moreover, this practice improves resource utilization by getting full trucks. All these advantages make cross-docking an interesting logistics practice for many companies. However, flows have to be well managed to keep these benefits achievable.

Keywords. Cross-dock, metaheuristics, local search, variable neighborhood search, variable neighborhood descent.

Univ. Artois, UR 3926, Laboratoire de Génie Informatique et d'Automatique de l'Artois (LGI2A), F-62400 Béthune, France.
*Corresponding author: cecilia.daquin@univ-artois.fr

Article published by EDP Sciences

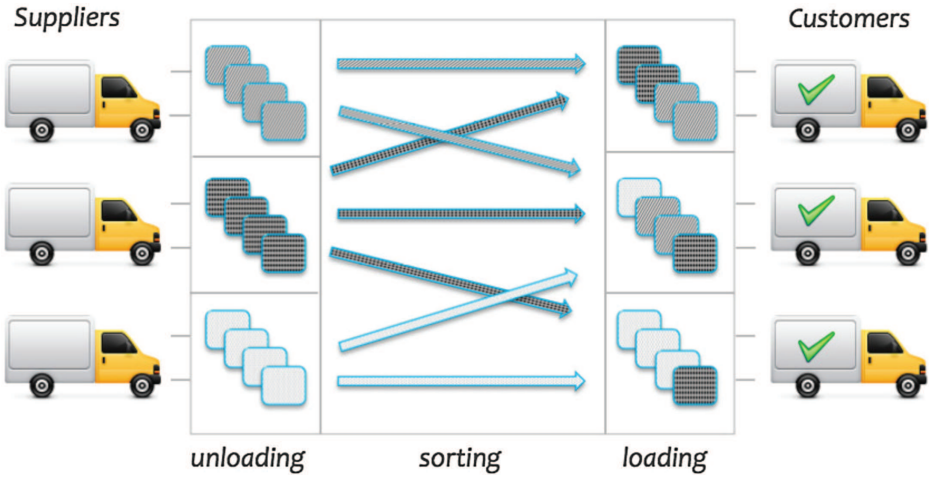

FIGURE 1. Schematic representation of a cross-dock.

Many decisions have to be taken by the practitioners about the use and implementation of cross-docking. These decisions have to be decided cautiously, because they have a major impact on the cross-docking efficiency. The first issue is to fix the location of the cross-dock and its layout. The network has to be designed so as to determine the goods flows through the network. Afterwards, the practitioners have to decide on routing of trucks before and after passing through the cross-dock. Once arriving at the cross-dock, the assignment of trucks to docks has to be addressed. Finally, resources inside the cross-dock have to be scheduled, and the storage area has to be located. In this paper, we focus on the assignment of trucks to docks in order to optimize the sum of operational cost and penalty cost. Because we assume that the number of trucks exceeds the number of docks, some penalties have to be paid when trucks cannot be scheduled in the current horizon. For this problem, we use the Integer Linear Programming (ILP) model proposed by Miao *et al.* [21] and propose an amendment to this model. According to our knowledge, this amended model was not considered in the literature yet. Since this combinatorial optimization problem is known to be NP-hard [21], we use several VNS variants to find optimal or near optimal solutions to our model.

The VNS algorithm consists of three steps which are executed until the stopping criterion is reached: the shaking step, the improvement procedure and the neighborhood change step (*cf.* Sect. 5). Following this basic scheme, different VNS variants can be obtained by choosing different settings: number and/or order of neighborhoods, the search strategy, the descent phase configuration, the stopping condition, etc. Thus, in this paper, we are interested in these different possible configurations of VNS in order to find the most effective for the studied cross-docking problem (Fig. 2). First, we focus on several neighborhood operators and the way they can be combined as VNS inputs to solve this problem. We propose four neighborhood operators. One of them is used by Lim *et al.* [20] and we develop the other ones for the cross-dock problem. We test and compare them one by one within a local search using two different search strategies. The aim is to highlight their characteristics and their efficiency in order to operate VNS algorithms much better as possible. Then, according to the results, we test several strategies for combining the operators in VNS structures. These strategies differ by the number and the order of the neighborhood operators. Regarding the VNS structure, we are interested in widely used variants of VNS method: the basic VNS (BVNS) that uses a local search in a descent phase, the general VNS (GVNS) that uses a basic VND (Variable Neighborhood Descent) and the GVNS that uses an union VND. After detecting the best strategy for the BVNS, we use it within the two GVNS variants and we compare the performances. Finally, we are interested in the impact of stopping criterion within VNS algorithm over the results. In each step of our empirical study, the performance of the tested algorithms is evaluated on a set of

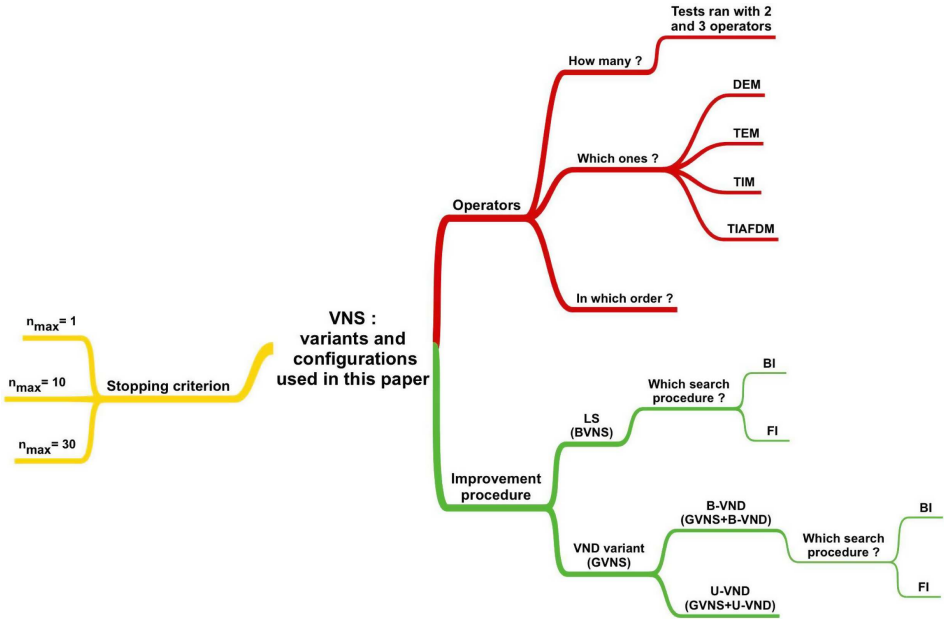

FIGURE 2. Overview of the VNS configurations tested in this paper.

instances of the cross-docking problem and compared with the performance of CPLEX solver. However, we couldn't compare our results with ones of Miao *et al.* [21] since their benchmarks are not available, and both models are slightly different (objective functions differ). Finally, the Friedman test is conducted to analyse the statistical significance of the proposed algorithms performance.

In the remainder of this paper, a literature review is presented in Section 2. Afterwards a formal description and the ILP model of the problem are found in Section 3. Sections 4 and 5 present the local search, the VNS variants used to solve this problem, their possible configurations and the proposed neighborhood operators. Computational results are given and analysed in Section 6. Finally, our results are summarized and possible future research is suggested in Section 7.

2. LITERATURE REVIEW

The cross-docking problem is well studied in recent literature. Boysen and Fliedner [6] provided a classification and a literature review for the truck scheduling problem. Van Belle *et al.* [35] presented a thorough review of papers about several problems related to cross-docking. Lately, Ladier and Alpan [17] compared the articles related to cross-docking operations found in the literature with the industrial practices captured through visits to cross-docks and interviews with their managers. Some papers discuss factors that influence the suitability of cross-docking. Apte and Viswanathan [3] examined features and techniques of effective cross-docking compared with traditional distribution. Van Belle *et al.* [35] provided guidelines for a successful implementation of cross-docking. Buijs *et al.* [7] proposed a framework specifying the interdependencies between different aspects of cross-docking problem and presented a new general classification scheme for cross-docking research based on the inputs and outputs for each problem aspect. Thus, during the design and operational phase of cross-docks, many decisions have to be taken.

In cross-docking, the temporary storage is an important element to manage. Indeed, products unloaded from trucks can be stored few hours at most while waiting for a truck to arrive. Thus, a staging strategy have to be decided. Taylor and Noble [32] examined three material staging alternatives in various cross-dock environments.

They suggested that staging needs can vary depending on different staging methods and the scenarios considering a range of demand types. Sandal [31] extended this study and examined several staging strategies in order to determine which one is the most appropriate in a cross-docking operation according to freight attributes and container loading requirements. Vis and Roodbergen [36] sought to determine temporary storage locations for incoming freight in order to minimize the travel distances of goods. Their method consisted of a network formulation that incorporated loading and unloading dock door locations, travel distances and available storage space in the facility. Experimental results showed the proposed algorithm can reduce the total travel distance up to about 40%.

One of the key decisions is the dock assignment for inbound and outbound trucks. An efficient dock assignment can increase the performance of the cross-dock and can reduce the shipment delays, the operational time and costs in the cross-dock, among other things. So, dock door assignment problem tries to provide the best assignment for trucks to docks at a given point in time. This problem considers the number of trucks being less than or equal to the number of docks, and the time dimension are not taken into consideration. The purpose is to assign each truck to a different door, considering the spatial dimension, *i.e.* the location of the doors. Tsui and Chang [33] formulated the dock door assignment problem as a bilinear programming problem where the objective function is nonlinear that tries to minimize the total distance traveled by the forklifts. A branch-and-bound algorithm is proposed to solve this problem. Cohen and Keren [9] suggested another optimal formation that is a non-linear MIP model and a heuristic approach for assigning cross-dock doors to trailers. Nassief *et al.* [28] formulated the cross-dock door assignment problem as a mixed integer programming which embedded into a Lagrangian relaxation that exploited the structure of the problem to obtain lower and upper bounds on the optimal solution. This approach allowed to obtain solutions of instances more difficult. Nassief *et al.* [29] proposed two new MIP models for this problem and compared the results of their computational and analytical experiments with existing ones.

Dock assignment problems do not take into consideration temporal constraints, so it is not possible to assign multiple trucks to one dock sequentially. Yet, other problems consider more trucks than docks and seek to determine the succession of the arriving and departing trucks at the docks. In this case, only the temporal dimension is taken into consideration, without spatial constraints. Then, the so-called truck scheduling problem considers dock doors as resources that have to be scheduled over time. Note that the dock assignment problem is part of the truck scheduling problem: over a first phase, the order and/or the time of trucks assignment is decided in order to choose a dock for each truck next. Chen and Lee [8] studied a two-machine cross-docking flow shop scheduling problem in which an operation on the second machine cannot be processed unless the operation on the first machine has been completed. Their purpose was to sequence both the inbound and outbound carriers to minimize the makespan. In their paper, they proved that the problem is strongly NP-hard and developed a branch-and-bound algorithm to solve it optimally. Yu and Egbelu [37] studied a cross-docking system where a temporary storage area is located in front of the shipping dock. They sought to find the best truck docking or best scheduling sequence for both inbound and outbound trucks to minimise the makespan. In their paper, a mixed integer programming model is formulated, and a heuristic algorithm is proposed to solve large problem instances. For the same problem, Vahdani and Zandieh [34] applied five metaheuristics to solve it including a genetic algorithm, tabu search, simulated annealing and variable neighborhood search. Their experimental study showed these metaheuristics resulted clearly better solution than the ones obtained by the heuristic in [37]. Boloori Arabani *et al.* [5] studied the same problem and developed also five metaheuristics: a genetic algorithm, tabu search, particle swarm optimization, ant colony optimization and differential evolution. Alpan *et al.* [2] proposed a bounded dynamic approach to schedule inbound and outbound trucks in a multiple receiving and shipping cross-dock environment. Alpan *et al.* [1] developed several heuristics to find the best schedule to minimize the sum of inventory holding and truck replacement costs. Lim *et al.* [20] considered a problem with time windows and capacity constraints, *i.e.* trucks are assigned to docks within its time window to unload and load the carried merchandises through the cross-dock with a limited capacity of storage. The objective was to minimize the total shipping distances between docks. In their paper, the problem is formulated as an integer programming model, then proposed a tabu search and a genetic algorithm to solve it. Later, they

extended this problem by focusing also on the operational time between docks [19]. Their new purpose was to minimize the operational cost, which is calculated according to the distance and the time travel between docks, and the penalty cost for unfulfilled shipments. This problem is formulated as an integer programming model and the authors proposed a genetic algorithm and finally a tabu search in [21] to solve it optimally. Miao *et al.* [22] proposed a new model in which the capacity constraint didn't take into consideration anymore, and distinguished the inbound trucks and docks from the outbound trucks and docks. They developed an adaptive tabu search to solve it. Gelareh *et al.* [15] proposed a different model for the same problem as in [21] and provided an efficient branch-and-cut algorithm to solve it in a reasonable computational time. Recently, Miao *et al.* [23] proposed a new model in which several practical constraints are simultaneously considered such as the capacity of trucks and of storage area, the time windows of trucks and the operational time within the crossdock. They proposed a two stage genetic algorithm to solve it. These last problems are similar to problems of gate assignment in airports. In the literature, some papers dealt with this subject [4,11,12,18,30]. Because the gate assignment problem is NP-hard [30] and it is a special case of the problem studied in [21], the latter showed their problem is NP-hard. Dondo and Cerdá [13] developed a mixed integer linear programming formulation for the vehicle routing problem with cross-docking to find the routing and scheduling of a mixed fleet, the truck docking sequence, the dock door assignment and the travel time to move the goods through the cross-dock. Mohtashami *et al.* [27] proposed a multi-objective mathematical model that minimizes the makespan, transportation costs and the number of truck trips in the entire supply chain. To solve it, they developed two evolutionary metaheuristic algorithms: the non-dominated sorting genetic algorithm and the multi objective particle swarm optimization. Fonseca *et al.* [14] proposed a lagrangian relaxation methodolody to solve a truck scheduling problem in a crossdock with parallel-docks. The objective is to minimize the makespan. Mohtashami [26] studied a cross-docking problem where there is temporary storage in front of the shipping dock and the shipping trucks can move in and out of the dock repeatedly. In fact, after the shipping truck loads some of its needed products, it can choose between two scenarios: either more products are loaded into the current shipping truck, or the current shipping truck is moved out from the shipping dock and another shipping truck is moved to the shipping dock to load its products. In this case, an outbound truck can load some of its needed products from shipping dock, leaves the dock for another outbound truck, waits and goes into the shipping dock again to load all or part of its remaining product items. Its purpose was to find the optimal/near optimal solution for sequencing inbound and outbound trucks to minimize the total operational cost. To solve this problem, the author developed a genetic algorithm-based method.

For the best of our knowledge, the proposed amendment to the model presented in Miao *et al.* [21] is considered for the first time. Since the problem is still NP-hard, it is unlikely to find optimal solutions for big instances in a reasonable computational time. It is why we propose in this paper several VNS metaheuristic variants to get approximate solutions. Hence, the main purpose here is to deliver a wide structured experimental study and a sensitivity analysis to compare several VNS configurations. The aim is to investigate some significant differences among them and to highlight the most effective for the studied problem.

3. Problem statement

Our study focuses on assignment of trucks to docks within a cross-dock under time window constraints as defined in [20,21]. Indeed, we seek to assign trucks to docks in order to minimize the sum of operational cost and penalty cost. The operational cost is the cost of transferring pallets from inbound docks to outbound docks. The penalty cost is the cost of not being able to transfer pallets from dock to dock. This is due to not assigning corresponding inbound truck or outbound truck in the required time window. In this case a negotiation could be engaged with the truck provider in the possibility to reassign it outside the required time window. We consider the problem whenever the number of trucks exceeds the number of available docks. The problem data are:

- The time window $[a_i; d_i]$ of each truck i, *i.e.*, a_i denotes the time instant in which truck i arrives and is assigned to a dock and d_i denotes the time instant in which truck i leaves its assigned dock. So then, the arrival and departure times of each truck have already been set;

- The cross-dock characteristics (number of docks, costs and time of transferring goods between docks and the maximum number of shipments that the warehouse can hold);
- The flow of shipments (the quantities and the penalty costs if goods are unfulfilled, the inbound and outbound trucks that have to transport goods).

Our purpose is to minimize the total cost, which is the sum of the cost of transferring goods between docks and the penalty cost. Therefore, we have to find the best dock assignments for trucks. Moreover, several constraints must be satisfied:

- A truck should be assigned to one dock during its time window, *i.e.* starting from its arriving time until its departure time. So then, it is necessary to ensure at least one dock is available during the time window of truck. If no dock is free, the latter will not be assigned and a penalty cost will be incurred for not transferred pallets;
- At any given time, only one truck can be assigned to a certain dock. If two trucks are assigned to the same dock, their respective time windows must not overlap;
- At any given time, the total number of goods inside the cross-dock cannot be greater than the maximum capacity of the warehouse (capacity constraint);
- The goods transfer process between two trucks has to take place between the arrival time of the inbound truck and the departure time of the outbound truck, while considering the time of transferring goods between two docks (precedence constraint).

Whenever a truck isn't assigned during its time window, a penalty cost is incurred for all the unfulfilled shipment by this truck.

Miao *et al.* [21] studied the same problem and proposed an IP model. We present in the following the same model except the objective for which we propose an amendment. Indeed, in the Miao *et al.*'s model, a transfer cost is incurred as soon as two docks are assigned, even if there is no shipment between these ones. We propose to add a new preprocessing variable σ in the operational cost to take into account an incurred cost if and only if there are transferred pallets between assigned trucks.

The following notations that we use are the same as in [21]:

N set of trucks arriving at and/or departing from the cross-dock.
M set of docks available in the cross-dock.
n total number of trucks.
m total number of docks.
a_i opening time of time window of truck i $(1 \leq i \leq n)$.
d_i closing time of time window of truck i $(1 \leq i \leq n)$.
$t_{k,l}$ operational time for pallets from dock k to dock l $(1 \leq k, l \leq m)$.
$f_{i,j}$ number of pallets moving from truck i to truck j $(1 \leq i, j \leq n)$.
$c_{k,l}$ operational cost per unit time from dock k to dock l $(1 \leq k, l \leq m)$.
$p_{i,j}$ penalty cost per unit shipment from truck i to truck j $(1 \leq i, j \leq n)$.
C capacity of cross-dock, *i.e.* the maximum number of shipment the cross-dock can hold at each time.

This binary pre-processing parameter is also defined:

$x_{i,j} = 1$ iff truck i departs no later than truck j arrives, *i.e.* iff $d_i \leq a_j$; 0 otherwise

We propose to add this new pre-processing parameter:

$\sigma_{i,j} = 1$ iff $f_{i,j} > 0$, *i.e.* if shipment has to be transferred from truck i to truck j; 0 otherwise.

The decision variables are as follows:

$$y_{i,k} = \begin{cases} 1 \text{ if truck } i \text{ is assigned to dock } k \\ 0 \text{ otherwise} \end{cases}$$

$$z_{i,j,k,l} = \begin{cases} 1 \text{ if truck } i \text{ is assigned to dock } k \text{ and truck } j \text{ is assigned to dock } l \\ 0 \text{ otherwise.} \end{cases}$$

Through a pre-processing step, we set the parameters $\sigma_{i,j}$ and $x_{i,j}$ $(1 \leq i, j \leq n)$. Moreover, in order to make data consistent, we give some assumptions as in [21]:

- $f_{i,j} \geq 0$, iff $d_j \geq a_i$ $(1 \leq i, j \leq n)$, otherwise $f_{i,j} = 0$. In other word, truck i will transfer some shipment to truck j iff truck j departs no earlier than truck i arrives. Note that even if $d_j \geq a_i$, $f_{i,j}$ can be equal to zero *i.e.* truck i has no shipment to transfer to truck j;
- $a_i < d_i$ $(1 \leq i, j \leq n)$ which means that for each truck, the arrival time must be strictly smaller than the departure time;
- $n > m$ because we consider the problem when the number of trucks exceeds the number of available docks;
- Capacity C is defined as follows: when truck i comes, then it will consume $\sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{j=1}^{n} f_{i,j}\ z_{i,j,k,l}$ units of capacity whenever i is assigned. Respectively, when truck j leaves its dock, then $\sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{i=1}^{n} f_{i,j}\ z_{i,j,k,l}$ units of capacity are released;
- Sort all the a_i and d_i in an increasing order $(1 \leq i, j \leq n)$, and let t_r (with r an integer such that $1 \leq r \leq 2n$) correspond to these $2n$ number such that $t_1 \leq t_2 \leq \ldots \leq t_{2n}$. Using these notations, we can easily formulate the capacity constraint afterwards.

In order to minimize the total cost, which is the sum of the cost of transferring goods and the penalty cost, we propose the following ILP model:

$$min \sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{k,l}\ t_{k,l}\ \sigma_{i,j}\ z_{i,j,k,l} + \sum_{i=1}^{n}\left(\sum_{j=1}^{n} p_{i,j}\ f_{i,j}\left(1 - \sum_{k=1}^{m}\sum_{l=1}^{m} z_{i,j,k,l}\right)\right) \tag{3.1}$$

s.t.

$$\sum_{k=1}^{m} y_{i,k} \leq 1\ (1 \leq i \leq n) \tag{3.2}$$

$$z_{i,j,k,l} \leq y_{i,k}\ (1 \leq i, j \leq n,\ 1 \leq k, l \leq m) \tag{3.3}$$

$$z_{i,j,k,l} \leq y_{j,l}\ (1 \leq i, j \leq n,\ 1 \leq k, l \leq m) \tag{3.4}$$

$$y_{i,k} + y_{j,l} - 1 \leq z_{i,j,k,l}\ (1 \leq i, j \leq n,\ 1 \leq k, l \leq m) \tag{3.5}$$

$$x_{i,j} + x_{j,i} \geq z_{i,j,k,k}\ (1 \leq i, j \leq n,\ i \neq j,\ 1 \leq k \leq m) \tag{3.6}$$

$$\sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{i\in\{i:a_i\leq t_r\}}\sum_{j=1}^{n} f_{i,j}\ z_{i,j,k,l} - \sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{i=1}^{n}\sum_{j\in\{j:d_j\leq t_r\}} f_{i,j}\ z_{i,j,k,l} \leq C\ (1 \leq r \leq 2n) \tag{3.7}$$

$$f_{i,j}\ z_{i,j,k,l}\ (d_j - a_i - t_{k,l}) \geq 0\ (1 \leq i, j \leq n,\ 1 \leq k, l \leq m) \tag{3.8}$$

$$y_{i,k} \in \{0,1\}, y_{j,l} \in \{0,1\}, z_{i,j,k,l} \in \{0,1\}\ (1 \leq i, j \leq n, 1 \leq k, l \leq m). \tag{3.9}$$

In this formulation, the objective function (3.1) is composed of two terms: the total operational cost and the total penalty cost. In the first part, the total operational cost represents the transfer cost between two docks according to the transfer time for each shipment transfer. Therefore, compared to [21], we add the binary parameter $\sigma_{i,j}$ in the definition of the total operational cost. Then, whenever $\sigma_{i,j} = 1$ *i.e.* when truck i has to transfer shipments to truck j, the transfer cost is applied if trucks i and j are assigned. If $\sigma_{i,j} = 0$, no transfer cost is added even if trucks i and j are assigned. In [21], this variable is not defined. In this case, with

constraint (3.5), a transfer cost is incurred as soon as two trucks are assigned, even if there is no shipment between theses ones. The second part corresponds to penalty incurred when goods are not transferred from dock to dock because corresponding trucks are not assigned or precedence constraints are not checked.

Constraint (3.2) ensures that each truck can be assigned to one dock at most. Constraints (3.3) to (3.5) jointly define the variable z which represent the logic relationship among $y_{i,k}$, $y_{j,l}$ and $z_{i,j,k,l}$. Constraint (3.6) guarantees that a dock cannot be occupied by more than one truck simultaneously. Constraint (3.7) defines the capacity constraint, *i.e.* at each event time (arrival and departure time of a truck), the total number of shipment inside the cross-dock cannot exceed the maximum capacity C. The last constraint (3.8) ensures that the shipment transfer between two trucks takes place within the time window of each truck while respecting precedence constraint between both of them.

4. Proposed local search algorithm

Before introducing the VNS framework, we first focus, in this section, on the neighborhood operators to analyse their efficiency within a local search. Local search (LS) is a simple procedure for heuristically solving combinatorial problems. This method is based on exploring the search space of candidate solutions until a "good" solution is found. The algorithm starts from an initial solution and then tries to "improve" it by moving to a neighborhood solution. A neighborhood structure $N(S,\ Op)$ is a set of solutions that differ only by one attribute or a combination of attributes from the current solution S. Neighborhood solutions are generated by applying a neighborhood operator Op to the current solution and a strategy of neighborhood progress.

The local search algorithm takes as parameters an initial solution S, a neighborhood operator and a strategy of neighborhood progress. Through each iteration, the neighbors of the current solution are created in accordance with the chosen neighborhood operator Op. Once a solution S' is identified from this neighborhood structure $N(S,\ Op)$, it is compared against the current solution S. If the candidate solution S' is better, it becomes the current solution and the search continues. The search stops whenever the stopping criterion is reached, *i.e.* whenever the current solution is getting stuck in a local optimum (no more improvement). To select a solution from the neighborhood, two strategies are used generally. The Best Improvement (BI) consists in choosing the best neighbor (the one with the lowest objective function). The First Improvement (FI) chooses the first neighbor which improves the current solution. Note that it is not necessary to generate all the neighbors of the current solution whenever the strategy FI is used.

The algorithm of local search is given in Algorithm 1.

We generate the initial solution with a "First Come First Served" rule (FCFS): trucks are sorted by their arrival time and are assigned to the first free dock in this order. If no dock is free for a given truck, the latter

Algorithm 1: Local search.

Function `localSearch`(S, Op, *strategy*)

Input : S: initial solution,
Op: neighborhood operator
strategy: BI or FI

Output: the solution S

repeat
  $N(S,\ Op) \leftarrow$ generate neighbors of S with operator Op;
  $S' \leftarrow$ choose a solution in $N(S,\ Op)$ according to *strategy*;
  if S' *better than* S **then**
    $S \leftarrow S'$;
  end
until *no more improvement*;
return S

FIGURE 3. Dock Exchange Move operator (DEM).

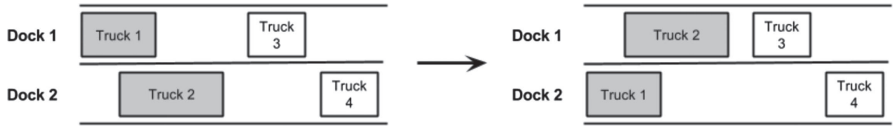

FIGURE 4. Truck Exchange Move operator(TEM).

will not be assigned. Hereafter, all the initial solutions used in the proposed algorithms are generated with this heuristic.

Neighborhood solutions are generated in accordance with a local search operator. In the rest of this study, the four operators presented below are used. Note that for each operator we examine all possible combinations.

Dock Exchange Move (DEM)

The first operator is a Dock Exchange Move (DEM): when considering two docks 1 and 2, all trucks assigned to dock 1 are moved to dock 2 and inversely. This operator enables to reduce the transferring costs between docks. Moreover, it only moves assigned trucks keeping their same time window and the same links of precedence as in the initial solution. Therefore, the produced solutions always satisfy all constraints of the problem. In the worst case, a solution can have $(m * m)/2$ neighbors, where m is the total number of docks (Fig 3).

Truck Exchange Move (TEM)

The second operator is a Truck Exchange Move (TEM): two trucks in a temporal conflict are exchanged. For example, let truck i be assigned to dock 1 and truck j assigned to dock 2. If both trucks are in a temporal conflict, truck i moves to dock 2 and truck j to dock 1. Two trucks are in a temporal conflict whenever their time windows have a shared part. The expected effects of this operator is to obtain larger free time windows (defragmentation) in order to receive new trucks. In addition, it enables to minimize transferring costs between docks. TEM can generate solutions that violate the time windows constraint. In this case, these invalid solutions are not added to the neighborhood. Thus, a solution can have $(n * m)$ neighbors in the worst case, where n is the total number of trucks and m the total number of docks (Fig 4).

Truck Insert Move (TIM)

The third operator is a Truck Insert Move (TIM): a truck i (assigned or not) is assigned to another dock. If there were trucks in this dock that are in conflict with it, then the last trucks are rejected. For example, truck i has to be assigned to dock 1. But the latter is taken over by truck j during the time windows of truck i. Accordingly, truck j is rejected to give a place to truck i. This operator enables to maximize the time windows, to reduce the transferring costs between docks and to minimize the penalty costs of non-delivering goods. TIM can generate solutions that break the time windows constraint and the capacity constraint of the cross-dock. Then, these invalid solutions are not added in the neighborhood. The neighborhood size will be equal to $(n*m)$ (Fig 5). This operator is used by Lim *et al.* [20].

Truck Insert At Free Dock Move (TIAFDM)

The last operator is a Truck Insert At Free Dock Move (TIAFDM): an assigned truck i is moved to a free dock during the time window of i. The expected effect is to fill the holes in dock allocations and to obtain larger

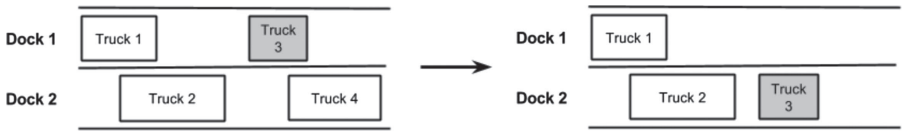

FIGURE 5. Truck Insert Move operator (TIM).

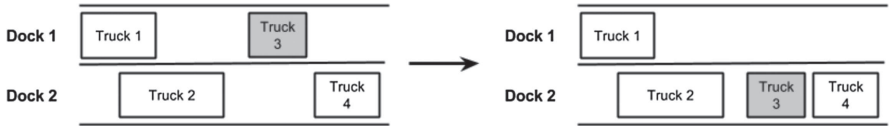

FIGURE 6. Truck Insert At Free Dock Move operator (TIAFDM).

time windows in order to receive a new truck. The solutions produced by this operator can violate the time windows constraint and the capacity constraint of the cross-dock. In this case, these invalid solutions aren't added in the neighborhood. In the worst case, this operator provides $(n * m)$ neighbors (Fig 6).

The main disadvantage of the local search is that the algorithm stops once the local optimum is reached and returns it. But, there is a big chance that a better solution exists in another neighborhood or in a larger neighborhood. Therefore, we only use this method to compare our proposed operators in order to combine it as better as possible in another metaheuristic: Variable Neighborhood Search (VNS).

5. Variable neighborhood search variants

VNS is a metaheuristic proposed by Mladenovìc and Hansen [25]. Its main characteristic is to change systematically the neighborhood during the optimisation process for an optimal (or near-optimal) solution. This procedure is performed in both methods: get a minimum local, and get out of the valley. VNS is developed upon the following observations:

- A local optimum with respect to one neighborhood structure is not necessarily a local optimum for another neighborhood structure;
- A global optimum is a local optimum with respect to all the neighborhood structures;
- For many problems, a large majority of local optima with respect to one or several neighborhoods are close to each other relatively.

The first property incites to move in several neighborhood in order to find local optima with respect to all the neighborhoods used. The second property suggests using several neighborhoods whenever local optima found are of poor quality. The last property means that a local optimum often provides some information about the global one and so the search for the vicinity of the current solution has to be intensified. Overall, the VNS algorithm repeats three steps until the stopping criterion is reached: the shaking step, the improvement procedure and then the neighborhood change step. The first step generates the neighbors of the current solution S within the current neighborhood operator and then a random solution S' is chosen among them. Thus, the shaking phase perturbs the local optimum found through the current iteration in order to provide a good starting point for the improvement procedure and get out of local optimum traps. Then, a procedure is applied starting from the selected solution S' in order to obtain a new local optimum S''. The final step allows to guide the search while exploring the solution space. More specifically, the neighborhood change step changes the neighborhood or continues in generating neighbors. Several neighborhood change procedures exist, but the widely used is sequential neighborhood step: if the local optimum S'' is better than the current solution S, then S'' becomes the current solution and the process starts again at the first step with the first neighborhood structure – otherwise, the search is resumed in the next neighborhood structure (according to a predefined order) of the

current solution S. The VNS algorithm stops whenever the stopping criterion is fulfilled. In this paper, the stopping criterion is reached whenever the current solution cannot be improved after testing all neighborhood structures in the course of a maximum number of iterations $n_{\max}$.

Many variants of VNS are deduced from this basic scheme [16]. In this paper, we focus on the widely used variants of VNS: basic VNS and general VNS.

Basic VNS (BVNS)

The BVNS parameters are: an initial solution (generated with a FCFS rule in this study) and a sorted list of neighborhood operators. Through each iteration, this algorithm executes the shaking step, the local search procedure (presented in Algorithm 1) and the neighborhood change step. These ones occur alternately until the current solution is not improved anymore in the course of a maximum number of iterations $n_{\max}$. The algorithm of the BVNS is given in Algorithm 2. In this strategy, S denotes the current solution which represents the best solution found at a given time. S' indicates a random solution used in the local search. The solution proposed by the local search S'' is a candidate solution that may become the current solution.

Algorithm 2: Basic Variable Neighborhood Search.

```
1  Function BVNS(S, Operators, strategy, n_max)
   Input  : S: initial solution,
            Operators: sorted list of κ_max neighborhood operators,
            strategy: BI or FI,
            n_max: stopping criterion
   Output: the solution S
1  nbIterationsWithoutImprovement ← 0; // number of consecutive iterations without improvement
2  repeat
3      κ ← 1;
4      while κ ≤ κ_max do
           // Shaking step
5          Op ← the κth operator in Operators;
6          N_κ ← generate neighbors of S with operator Op;
7          S' ← choose a random solution in N_κ;
           // Improvement procedure: local search
8          S'' ← localSearch(S', Op, strategy);
           // Sequential Neighborhood change step
9          if the local optimum S'' is better than S then
10             S ← S'';
11             κ ← 1; // return to the first neighborhood
12         else
13             κ ← κ + 1; // move to the next neighborhood
14     end
15 end
16 nbIterationsWithoutImprovement++;
17 until nbIterationsWithoutImprovement == n_max;
18 return S
```

General VNS (GVNS)

Contrary to the BVNS, the GVNS uses a Variable Neighborhood Descent (VND) procedure instead of a local search. This function requires as parameters an initial solution S (generated with a FCFS rule in this study) and two sorted lists of neighborhood operators. One of them is used within the shaking step and the other one within the VND procedure. To simplify the study hereafter, these two sorted lists will be the same.

The algorithm of the GVNS is given in Algorithm 3. Several variants of VND exist in the literature and some of them were compared in [24]. In this study, two procedures stood out from the others: the basic sequential VND (B-VND) and the union VND (U-VND). In our paper, we focused on this two variants. The parameters of these two functions are an initial solution S and a sorted list of neighborhood operators. The B-VND algorithm explores iteratively each neighborhood defined by the current operator according to the established order. The search strategy BI or FI is used to select a solution within a neighborhood. Whenever the current solution is improved, the process starts again at the first neighborhood structure. The algorithm is stopped if the current solution cannot be improved. The U-VND process is the same as the B-VND except that the search is continued in a single neighborhood that is obtained as the union of all predefined neighborhoods. This algorithm is used with BI search strategy, else U-VND is equivalent with B-VND. The algorithms of B-VND and U-VND are given in Algorithm 4 and Algorithm 5 respectively.

Algorithm 3: General Variable Neighborhood Search.

Function `GVNS`(S, *Operators_shaking*, *Operators_VND*, *strategy*, $n_{\max}$)

Input : S: initial solution,
Operators_shaking: sorted list of $\kappa_{\max}$ neighborhood operators for the shaking step,
Operators_VND: sorted list of $\lambda_{\max}$ neighborhood operators for the VND procedure,
strategy: BI or FI
$n_{\max}$: stopping criterion

Output: the solution S

$nbIterationsWithoutImprovement \leftarrow 0$;
repeat
  $\kappa \leftarrow 1$;
  while $\kappa \leq \kappa_{\max}$ **do**
    `// Shaking step`
    $Op \leftarrow$ the κth operator in *Operators_shaking*;
    $N_\kappa \leftarrow$ generate neighbors of S with operator Op;
    $S' \leftarrow$ choose a random solution in N_κ;
    `// Improvement procedure:` VND `is either` $B-VND$ `or` $U-VND$
    `// in this study`
    $S'' \leftarrow VND(S',$ *Operators_VND*, *strategy*$)$;
    `// Sequential Neighborhood change step`
    if S'' *is better than* S **then**
      $S \leftarrow S''$;
      $\kappa \leftarrow 1$; `// return to the first neighborhood`
    else
      $\kappa \leftarrow \kappa + 1$; `// move to the next neighborhood`
    end
  end
  $nbIterationsWithoutImprovement++$;
until $nbIterationsWithoutImprovement == n_{\max}$;
return S;

6. Experimental study

According to the previous Sections 4 and 5, the VNS algorithms can differ depending on the following configuration aspects (Fig. 2):

Algorithm 4: Basic sequential Variable Neighborhood Descent.

Function `B-VND`(*S, Operators, strategy*)
Input : S: initial solution,
Operators: sorted list of $\lambda_{\max}$ neighborhood operators
strategy: BI or FI
Output: the solution S

repeat
  $\lambda \leftarrow 1$;
  while $\lambda \leq \lambda_{\max}$ **do**
    $Op \leftarrow$ the λth operator in *Operators*;
    $N_\lambda \leftarrow$ generate neighbors of S with operator Op;
    $S' \leftarrow$ select a solution in N_λ according to *strategy*;
    `// Sequential Neighborhood change step`
    if S' *is better than* S **then**
      $S \leftarrow S'$;
      $\lambda \leftarrow 1$; `// return to the first neighborhood`
    else
      $\lambda \leftarrow \lambda + 1$; `// move to the next neighborhood`
    end
  end
until *no more improvement for* S $(S \leq S')$;
return S;

Algorithm 5: Union Variable Neighborhood Descent.

Function `U-VND`(*S, Operators, strategy*)
Input : S: initial solution,
Operators: sorted list of $\lambda_{\max}$ neighborhood operators
strategy: BI
Output: the solution S

repeat
  $N \leftarrow N_1 \cup N_2 \cup \ldots \cup N_{\lambda_{\max}}$;
  $S' \leftarrow$ select the best solution in N; `// BI strategy`
  if S' *is better than* S **then**
    $S \leftarrow S'$;
  end
until *no more improvement for* S $(S \leq S')$;
return S;

(1) Neighborhood operators: among the proposed operators DEM, TEM, TIM and TIAFDM, which one(s) to use in VNS algorithms? How many? In which order?
(2) Search strategy for the improvement procedure: first (FI) or best improvement (BI)?
(3) Improvement procedure: a simple local search or a VND variant (B-VND or U-VND)? In other words, which VNS variants to be used: BVNS, GVNS using B-VND or GVNS using U-VND?
(4) Stopping criterion: what is the necessary amount of consecutive iterations while the solution remains without improvement? 1, 10 or 30 iterations?

We can see that many different VNS variants can be obtained by choosing different settings. We can also think that these choices could have an impact on the quality of the final solution. Hence in the rest of this paper, we conduct several experiments to compare several VNS metaheuristic variants under these different settings.

The aim of this empirical study is to find the most effective configuration(s) of VNS metaheuristic for solving the cross-dock problem.

This experimental study consists of four parts. In the first part, we evaluate and compare the four neighborhood operators one by one within a local search. These tests are conducted under two different settings: a local search that uses BI search strategy and a local search that uses FI search strategy. The second part is devoted to the way these operators could be combined in a BVNS structure according to their efficiency. We test several strategies that differ by the number and the order of the neighborhood operators. Then, a comparison of VNS variants is performed in the third part. In the last part, we evaluate the influence of the stopping criterion within these variants over the results. In each part of this study, the performance of the tested algorithms is evaluated on various instances of the cross-docking problem. The best performing algorithms are compared with the performance of CPLEX solver. A detailed description of the results is presented in the tables of Appendix A. These results are also visible on the following link https://www.lgi2a.univ-artois.fr/~hsu/Recherche/CROSSDOCK-BENCHS/.

6.1. Experimental setup and parameter settings

The computational experiments are conducted on Intel ®Core (™) i7 CPU K875 @ 2.93 GHz of a machine using 2.00 Gb RAM. The runs are made on a 64 bits machine. The instances used are generated in the same way as in [21] and they are sorted by the number of trucks and also the number of docks. In the first row of each table of results in Appendix A, $n \times m$ denotes that there are n trucks and m docks for a group of instances. Each category consists of five instances. In total we have 95 instances (19 groups of instances) ranging from $n = 10$ and $m = 3$ up to $n = 60$ and $m = 10$. In our study, each instance has been executed thirty five times and each result cell in the table contains the average value of these runs. As comparison, we use ILOG CPLEX 12.6.3 with default settings to solve the formulation presented in Section 3 with a time limit of 7200 s. The solution quality and computational time of the proposed metaheuristics are evaluated. Hereafter, whenever comparing two methods, the mentioned percentages of improvement of the solution quality (respectively runtime) are determined firstly by evaluating for each instance the ratio between the value of objective function (or runtime) obtained with the first method and the value obtained with the second method. Afterward, we calculate the average of all the objective function ratios (or runtime) for all the instances. To evaluate the statistical significance of our results, the Friedman test, based on [10] has been conducted.

The Friedman test is a non-parametric statistical test used to detect significant differences between the behavior of k ($k \geq 2$) algorithms through a set of b instances. To perform the Friedman test, the first step is to rank the results X_{ij} of the metaheuristics j ($j \leq k$) for each instances i ($i \leq b$) separately. Let $R(X_{ij})$ be the rank from 1 to the best performing algorithm and k to the worst. Then, the total summation of squared ranks A_2 is calculated:

$$A_2 = \sum_{i=1}^{b} \sum_{j=1}^{k} [R(X_{ij})]^2 . \tag{6.1}$$

Therefore, for each metaheuristic j ($j \leq k$) the summation of the rank is computed $R_j = \sum_{i=1}^{b} R(X_{ij})$, then B_2 is calculated:

$$B_2 = \frac{1}{b} \sum_{j=1}^{k} {R_j}^2. \tag{6.2}$$

In this test, under the null-hypothesis, which states that all the algorithms behave similarly, the Friedman statistic can be computed as:

$$T_2 = \frac{(b-1)\left[B_2 - bk(k+1)^2/4\right]}{A_2 - B_2} \tag{6.3}$$

which is distributed according to an F distribution with $(k-1)$ and $(k-1)(b-1)$ degrees of freedom. If T_2 is greater than $1-\alpha$ quantile of the F distribution with $(k-1)$ and $(k-1)(b-1)$ degrees of freedom

(α a significance level), the null-hypothesis is rejected, *i.e.* the existence significant differences is found. In this case, we proceed with a post-hoc procedure to characterize these differences.

The post hoc test is used to compare algorithms two by two. This test declares the algorithms i and j significantly different if the absolute difference between the summation of the ranks for i and for j is greater than the critical value for the difference with $(b-1)(k-1)$ degrees of freedom and α a significance level.

In this study, we always use a significance level α equals to 0.01 for the Friedman test and the post-hoc tests will be employed for a significant degree $\alpha = 0.05$.

6.2. Results

6.2.1. Comparison of neighborhood operator within LS

In this first part, we evaluate one by one the neighborhood operators (DEM, TEM, TIM and TIAFDM presented in Sect. 4) within a local search. For each operator, LS is tested under two different settings: BI and FI search strategies. The purpose is to determine the efficiency of each operator and the influence of the search strategies in term of solution quality and runtime. The results are presented in Tables A.1 and A.2

Let's focus on the search strategies BI and FI. For all operators, considering the average results, we note that BI provides the best and fastest solutions. Because FI picks the first improvement in the neighborhood which can be a solution far away from the best neighbor. Consequently, the local search algorithm is slower in finding the local optima.

Regarding the neighborhood operators with BI strategy, we can deduce:

(1) If we consider the average solutions (see the average results over entire set of instances in column *Average* of Tab. A.2), TIM provides the best solution quality, DEM is ranked second followed by TEM, and TIAFDM is the worst. This ranking is the same for each group of instances (except for 14×4 and 14×6).
(2) Regarding the runtime consumed by each operator, if we consider the average solutions, we obtain the following ranking: TIAFDM is the fastest, DEM and TEM are second and third respectively, and TIM is the slowest. This ranking is the same for each group of instances (except for 10×3).
(3) For each group of instances, TIM is the best operator in term of solution quality, but it is slower than the others. In fact, contrary to the other operators, TIM tries to move all trucks (assigned or not) to another dock available or not at the risk of rejecting another truck. Hence, this operator is slower, but it improves the solution quality because the penalty cost can be minimized by trying to assign rejected trucks.
(4) For each group of instances, TIAFDM is the worst operator in term of solution quality, but it is the faster. This behavior is as expected because usually TIAFDM provide a smaller neighborhood than the others and with no many improved solutions. So the search is faster and with lower quality.

To highlight the significant differences between these neighborhood operators, we conduct the Friedman test. Table 1 shows the results of the Friedman test for solution quality and runtime and summarizes the ranking obtained. In this table and the following ones reporting the results of Friedman test, "Average rank" notices the average of ranks obtained by the corresponding metaheuristic according to the solution quality (or the runtime) over the b groups of instances. Similarly, "Summation" notices the sum of the b ranks obtained and of the b squared ranks. We can notice on Table 1 that for both solution quality and runtime, $T_2 > F_{0.99,3,54}$ *i.e.* the null hypothesis is rejected. It means that there exists at least one operator for the local search whose performance is different from at least one of other operators. Consequently the post-hoc paired comparisons is performed to know which neighborhood operator outperforms others.

Table 2 shows the results of the post-hoc test for solution quality and runtime. Each cell of this table contains the absolute difference between the summation of the ranks for the operator of the line (indicated in the Tab. 1) and the summation of the ranks for the operator of the column. Symbol "$(-)$" indicates that the result of the subtraction is negative, that means that the operator of the line provides better performances than the operator of the column. Conversely, the absence of this symbol indicates that the operator of the column outperforms the one of the line. Underlines values indicate the operators are significantly different because these values are greater than the critical value. For example, let's compare the operators DEM and TEM in term of solution

TABLE 1. Results of the Friedman test for solution quality and runtime of the Local Search (with $k = 4$, $b = 19$ and $\alpha = 0.01$).

	DEM		TEM		TIM		TIAFDM	
	$R(X_{b_1})$	$R(X_{b_1})^2$	$R(X_{b_2})$	$R(X_{b_2})^2$	$R(X_{b_3})$	$R(X_{b_3})^2$	$R(X_{b_4})$	$R(X_{b_4})^2$
Solution quality								
Average rank	2.11		2.89		1		4	
Summation	40	86	55	161	19	19	76	304
Runtime								
Average rank	2.05		2.95		4		1	
Summation	39	81	56	166	76	304	19	19
	Solution quality						Runtime	
T_2	459.79						884.5	
$F_{0.99,3,54}$				4.17				

TABLE 2. Results of the post-hoc test for solution quality and runtime of the Local Search (with $\alpha = 0.05$ and 54 degrees of liberty).

(a) Solution quality				
	DEM	TEM	TIM	TIAFDM
DEM	–	15 (–)	21	36 (–)
TEM	–	–	36	21 (–)
TIM	–	–	–	57 (–)
TIAFDM	–	–	–	–
Critical value			20.99	
(b) Runtime				
	DEM	TEM	TIM	TIAFDM
DEM	–	19 (–)	38 (–)	19
TEM	–	–	19 (–)	38
TIM	–	–	–	57
TIAFDM	–	–	–	–
Critical value			20.99	

quality: $40-55 = -15$; hence the absolute value "15" following by "(−)" in the cell of the line DEM and column TEM; as $15 < 20.99$ (the critical value), DEM and TEM have the same performance. Let's compare the quality solution of DEM and TIAFDM now, *i.e.* the last cell of the line DEM of the table: $40 - 76 = -36$; $36 > 20.99$ so DEM and TIAFDM are significantly different; as the subtraction give a negative result, DEM outperforms TIAFDM. In this way, we can notice that TIM outperforms significantly each one of the other operators in term of solution quality, contrary to TIAFDM which is significantly outperformed by all the other operators. In term of runtime, DEM is equivalent to TEM and to TIAFDM. DEM is faster than TIM and TIAFDM outperforms TEM and TIM.

According to these tests and analysis, we can conclude this following ranking of neighborhood operators: TIM is the best operator, followed jointly by DEM and TEM, TIAFDM takes the last place. Even if TIM is the slowest, it needs in average less of 3 s to propose a solution 26% better that the one proposed by TIAFDM.

6.2.2. Neighborhood operators within BVNS

This second part is designed to answer to the following questions: how many neighborhood operators is better to use within a basic VNS? In which order? With this in mind, we evaluate BVNS first with two operators and then with three operators. During these tests, we use the three best operators according to the previous part: TIM, DEM and TEM. Also, BVNS algorithm uses BI for the search strategy and stops when the solution cannot be improved, *i.e.* with $n_{\max} = 1$.

Firstly we evaluate BVNS with these six possible orders of the neighborhoods:

- TEM-TIM
- TIM-TEM
- DEM-TIM
- TIM-DEM
- TEM-DEM
- DEM-TEM

The results are presented in Tables A.3 and A.4. Let's focus on the order of the operators. We can notice this following observations:

(1) TEM-TIM *versus* TIM-TEM: on average solution (see the average results over entire set of instances in column *Average* of Table A.4) and for each group of instances, TIM-TEM provides better solution quality (except for 60×10). Regarding the runtime on the average results, TEM-TIM is slightly better.
(2) DEM-TIM *versus* TIM-DEM: DEM-TIM provides provides better solution quality for each group of instances (except for 18×6) but it is slower (except for 14×4). This observation is valid on the average solution.
(3) TEM-DEM *versus* DEM-TEM: if we consider average results, DEM-TEM is better regarding the solution quality and the runtime. Also, this order is faster for each group of instances (except for 16×6 and 25×6).

The results of Friedman tests are summarized in Table 3. Obtained tests show that the null hypothesis is rejected for the solution quality and the runtime as well. The results of the post-hoc test is presented in Table 4. Comparing the order of the operators, the performance is the same for solution quality and runtime, except TIM-TEM provides better solution than the reversed order.

According to this analysis, we can conclude that TIM-TEM outperforms the reversed order, just like TIM-DEM and DEM-TEM.

When we compare the six ordered lists of operators, if we consider the average results (column *Average* of Tab. A.4), we can notice:

(1) Regarding the solution quality, we obtain the following ranking: TIM-TEM provides the best solution, followed by TIM-DEM, then TEM-TIM and DEM-TIM, the two last places are assigned to DEM-TEM and finally TEM-DEM.
(2) For each group of instances, the best solution is provided by the ordered lists with TIM as the first operator (except for 18×6 and 60×10).
(3) Comparing the runtime, the fastest is DEM-TEM followed by the reversed order (TEM-DEM), DEM-TIM and the reversed order are ranked third and fourth respectively, TEM-TIM and TIM-TEM are the slowest ranked in the fifth and the last place respectively.

The statistical tests show that TIM-TEM provides better solutions than the other operators. DEM+TEM (in two-ways) is worse than all the other operators in term of solution quality, but is the fastest.

Actually, based on these observations, the behavior of the operator TIM shows that it is a typical intensification operator. Indeed the intensification seeks to manage the search in the area around the best found combinations by favouring through each step the choice of solutions that belong to best neighborhoods built earlier. Hence, because TIM provides on average the best solution quality combining with the LS, it is able to explore the search space. As a second operator which is the exploration/diversification operator, DEM and TEM perform better, even if TEM seems to be slightly better. Actually, they are able to diversify the solution in

TABLE 3. Results of the Friedman test for solution quality and runtime of basic VNS with two operators (with $k = 6$, $b = 19$ and $\alpha = 0.01$).

	DEM-TIM		TIM-DEM		TEM-TIM	
	$R(X_{b_1})$	$R(X_{b_1})^2$	$R(X_{b_2})$	$R(X_{b_2})^2$	$R(X_{b_3})$	$R(X_{b_3})^2$
Solution quality						
Average rank	3.21		1.95		3.42	
Summation	61	205	37	81	65	239
Runtime						
Average rank	3.05		4.05		5.53	
Summation	58	178	77	317	105	585
	TIM-TEM		TEM-DEM		DEM-TEM	
	$R(X_{b_4})$	$R(X_{b_4})^2$	$R(X_{b_5})$	$R(X_{b_5})^2$	$R(X_{b_6})$	$R(X_{b_6})^2$
Solution quality						
Average rank	1.42		5.53		5.47	
Summation	27	45	105	585	104	574
Runtime						
Average rank	5.37		1.89		1.11	
Summation	102	554	36	70	21	25
	Solution quality				Runtime	
T_2	99.72				272.09	
$F_{0.99,5,90}$			3.23			

TABLE 4. Results of the post-hoc test for solution quality and runtime of basic VNS with two operators (with $\alpha = 0.05$ and 90 degrees of liberty).

(a) Solution quality						
	DEM-TIM	TIM-DEM	TEM-TIM	TIM-TEM	TEM-DEM	DEM-TEM
DEM-TIM	–	24	4 (–)	34	44 (–)	43 (–)
TIM-DEM	–	–	28 (–)	10	68 (–)	67 (–)
TEM-TIM	–	–	–	38	40 (–)	39 (–)
TIM-TEM	–	–	–	–	78 (–)	77 (–)
TEM-DEM	–	–	–	–	–	1
DEM-TEM	–	–	–	–	–	–
Critical value			33.85			
(b) Runtime						
	DEM-TIM	TIM-DEM	TEM-TIM	TIM-TEM	TEM-DEM	DEM-TEM
DEM-TIM	–	19 (–)	47 (-)	44 (–)	22	37
TIM-DEM	–	–	28 (–)	25 (–)	41	56
TEM-TIM	–	–	–	3	69	84
TIM-TEM	–	–	–	–	66	81
TEM-DEM	–	–	–	–	–	15
DEM-TEM	–	–	–	–	–	–
Critical value			33.85			

TABLE 5. Results of the Friedman test for solution quality and runtime of the VNS algorithms with three operators (with $k = 4$, $b = 19$ and $\alpha = 0.01$).

	BVNS_ TIMTEMDEM		BVNS_ TIMDEMTEM		GVNS+ B-VND		GVNS+ U-VND	
	$R(X_{b_1})$	$R(X_{b_1})^2$	$R(X_{b_2})$	$R(X_{b_2})^2$	$R(X_{b_3})$	$R(X_{b_3})^2$	$R(X_{b_4})$	$R(X_{b_4})^2$
Solution quality								
Average rank	2.58		2.74		1.21		3.47	
Summation	49	141	52	150	23	33	66	246
Runtime								
Average rank	1.74		1.32		4		2.95	
Summation	33	63	25	37	76	304	56	166
	Solution quality						Runtime	
T_2			20.68				141.26	
$F_{0.99,3,54}$				4.17				

order to escape local optima and to explore the search space because of the large neighborhood size that TEM and DEM provide. Finally, we can also conclude that TIM-TEM is the best combinaison for BVNS. Hereafter, the latter will be called BVNS_TIMTEM.

In the second step of this part, we decide to evaluate the BVNS algorithm with three operators. Because we showed that TIM is an intensification operator previously, TIM is placed as the first operator. Hence, we compare BVNS using the ordered lists TIM-TEM-DEM (hereafter called BVNS_TIMTEMDEM) and TIM-DEM-TEM (called BVNS_TIMDEMTEM). The results are presented in Tables A.5 and A.6. If we consider the average results (see column *Average* of Tab. A.6), we can notice:

(1) BVNS_TIMTEMDEM provides better solution quality than BVNS_TIMDEMTEM but it is slightly slower. The statistical analysis shows that BVNS_TIMTEMDEM and BVNS_TIMDEMTEM have the same performance in term of both solution quality and runtime (Tabs. 5 and 6).
(2) When we compare with BVNS_TIMTEM, the last one provides the worst solution quality but it is faster than BVNS using three operators. In fact, adding an operator in BVNS increases the runtime of the algorithm, but allows exploring further the search space.

We can conclude that the list TIM-TEM-DEM in this order is the best sequence for BVNS among all those tested. Even if this one is the slowest, it needs in average less of 14 s to perform.

6.2.3. Comparaison with GVNS

In this third experimental part, we combine the operators TIM-TEM-DEM in this order in another variant of the VNS algorithm: the GVNS. First, we use GVNS with the B-VND (hereafter called GVNS+B-VND), then GVNS is combined with the U-VND (hereafter called GVNS+U-VND). In these two scenarios, GVNS and VND algorithms take as parameters the same list of neighborhood operators (TIM-TEM-DEM). Like BVNS in the previous experiments, GVNS algorithm stops when the solution cannot be improved, *i.e.* with $n_{\max} = 1$.

From the results presented in Tables A.5 and A.6, we can notice if we consider the average results (see column *Average* of Tab. A.6):

(1) GVNS+B-VND provides better solution quality than GVNS+U-VND but it is slower. This observation is valid for each group of instances (except for the solution quality in 16×4). The good quality of GVNS+B-VND regarding to GVNS+U-VND is due to the fact that union VND algorithm's policy is to seek a local optimum within a same large neighborhood obtained here as the union of three neighborhoods. Thus, it can be stuck in local optima.

TABLE 6. Results of the post-hoc test for solution quality and runtime of the VNS algorithms with three operators (with $\alpha = 0.05$ and 54 degrees of liberty).

(a) Solution quality

	BVNS_ TIMTEMDEM	BVNS_ TIMDEMTEM	GVNS+ B-VND	GVNS+ U-VND
BVNS_TIMTEMDEM	–	3 (–)	26	17 (–)
BVNS_TIMDEMTEM	–	–	29	14 (–)
GVNS+B-VND	–	–	–	43 (–)
GVNS+U-VND	–	–	–	–
Critical value		20.99		

	BVNS_ TIMTEMDEM	BVNS_ TIMDEMTEM	GVNS+ B-VND	GVNS+ U-VND
BVNS_TIMTEMDEM	–	8	43 (–)	23 (–)
BVNS_TIMDEMTEM	–	–	51 (–)	31 (–)
GVNS+B-VND	–	–	–	20
GVNS+U-VND	–	–	–	–
Critical value		20.99		

(2) Comparing GVNS variants and BVNS on the solution quality, we obtain the following ranking: GVNS+B-VND provides the best solution quality, BVNS_TIMTEMDEM is ranking second, and GVNS+U-VND is the worst. GVNS+U-VND is the worst because at each iteration, a local optimum is generated with the respect to three neighborhoods structure together, whereas BVNS_TIMTEMDEM uses one operator only that can be different at a next iteration. So the last one has higher probability to get out of a local optima. In the same way and because GVNS+B-VND uses B-VND, it enables to find a local optimum that is more likely to be a global optimum.

(3) Regarding the runtime, the ranking is as follows: BVNS_TIMTEMDEM is the fastest, the second is GVNS+U-VND and the slowest is GVNS+B-VND. In the same way, using several neighborhood structures to search a local optimum consumes more time than with a single neighborhood structure.

The statistical analysis validates these observations. Table 5 summarizes Friedman results for solution quality and runtime. In the two cases, the null hypothesis is rejected. The post-hoc results (see Tab. 6) show that GVNS+B-VND is better in term of solution quality than GVNS+U-VND. GVNS+B-VND outperforms each one of the other variants in term of solution quality, but GVNS+B-VND and GVNS+U-VND are the slowest.

In conclusion, GVNS+B-VND using TIM-TEM-DEM as ordered list provides the best solution quality, but it is the slowest (it is 2 times slower than BVNS_ TIMTEMDEM). However, the runtime consumed by this algorithm is still reasonable because it needs in average 28 s to perform. So, all the previous experiments lead to the ranking for the three best strategies: GVNS+B-VND is at the head, followed by BVNS_TIMTEMDEM, while GVNS+U-VND is ranking as third.

6.2.4. Influence of the stopping criterion

The GVNS algorithm described in Section 5 stops whenever the solution cannot be improved through $n_{\max}$ iterations. In our analysis above, we chose to set $n_{\max}$ to 1. In this last part of the tests, we evaluate the influence of this parameter on the solution quality by increasing the value of $n_{\max}$. To realize this experiment, we use our best metaheuristic GNVS+B-VND until now and we compare the results obtained with $n_{\max}$ set to 1, then to 10 (called GNVS+B-VND_10) and to 30 (GNVS+B-VND_30).

From the results presented in Tables A.5 and A.6, we can see:

(1) Regarding the solution quality, for each instance we obtain always the followed ranking: GNVS+B-VND_30 is the best, GNVS+B-VND_10 is ranked in second, while GNVS+B-VND is the worst.

TABLE 7. Results of the Friedman test for solution quality of the VNS algorithms with $n_{\max} \in \{1; 10; 30\}$ (with $k = 3$, $b = 19$ and $\alpha = 0.01$).

	GVNS+B-VND		GVNS+B-VND_10		GVNS+B-VND_30	
	$R(X_{b_1})$	$R(X_{b_1})^2$	$R(X_{b_2})$	$R(X_{b_2})^2$	$R(X_{b_3})$	$R(X_{b_3})^2$
Average rank	3		1.97		1.03	
Summation	57	171	37.5	74.25	19.5	19.5
$F_{0.99,2,36}$			5.27			

TABLE 8. Results of the post-hoc test for solution quality of the VNS algorithms with $n_{\max} \in \{1; 10; 30\}$ (with $\alpha = 0.05$ and 36 degrees of liberty).

	GVNS+B-VND	GVNS+B-VND_10	GVNS+B-VND_30
GVNS+B-VND	–	19.5	37.5
GVNS+B-VND_10	–	–	18
GVNS+B-VND_30	–	–	–
Critical value		14.76	

(2) Regarding the runtime, for each instance, the ranking is in the reversed order than the previous one: GNVS+B-VND is the fastest, GNVS+B-VND_10 stays in second place, and GNVS+B-VND_30 is the slowest.

Friedman test is applied to the results of these three variants. The results are summarized in Table 7. In term of solution quality, the null hypothesis is rejected. The post-hoc analysis (see Tab. 8) shows that GNVS+B-VND is outperformed by the other two, but in particular GNVS+B-VND_30 provides the best solutions. These results can be explained easily. The more the GVNS algorithm continues its search, the more it can escape the local optima thanks to the shaking step and explore the search space. So, the algorithm has more opportunity to find the optimal solution. However, the procedure surely consumes more CPU times. In our study, with $n_{\max}$ set to 30, GVNS takes in average less than 5 min (less than 1 s for small instances and up to 40 min for the biggest instances). To solve the cross-dock issue as an operational problem, this runtime is still reasonable.

Figure 7 describes the distribution of studied algorithms according to their performance. The values are the total of the objective functions and the total runtime for all instances. Using the Pareto front, non dominated approaches can be deduced.

6.3. Comparison with exact method

Now that we compared different possible configurations for VNS variants to solve the cross-dock problem, we evaluate the performances of the best algorithms (*i.e.* BVNS_TIMTEMDEM, GVNS+B-VND variants and GVNS+U-VND) with those of CPLEX configured with a time limit of 7200 s.

Let's focus on small instances (from 10×3 to 18×4). We can observe:

(1) CPLEX always finds optimal solution in less than two hours. However, for each group of instances, CPLEX is slower than the VNS variants.

(2) Our best VNS variants can get near optimal solutions in a much shorter time than CPLEX. Table 9, shows the average gap expressed in percentage of the objective function values between the best algorithms and CPLEX for the small instances. Each percentage is calculated by averaging the gaps for the small instances presented in the tables of Appendix A for the corresponding algorithm. This gap is less than 1%, and goes down 0.3% with GVNS+B-VND_30.

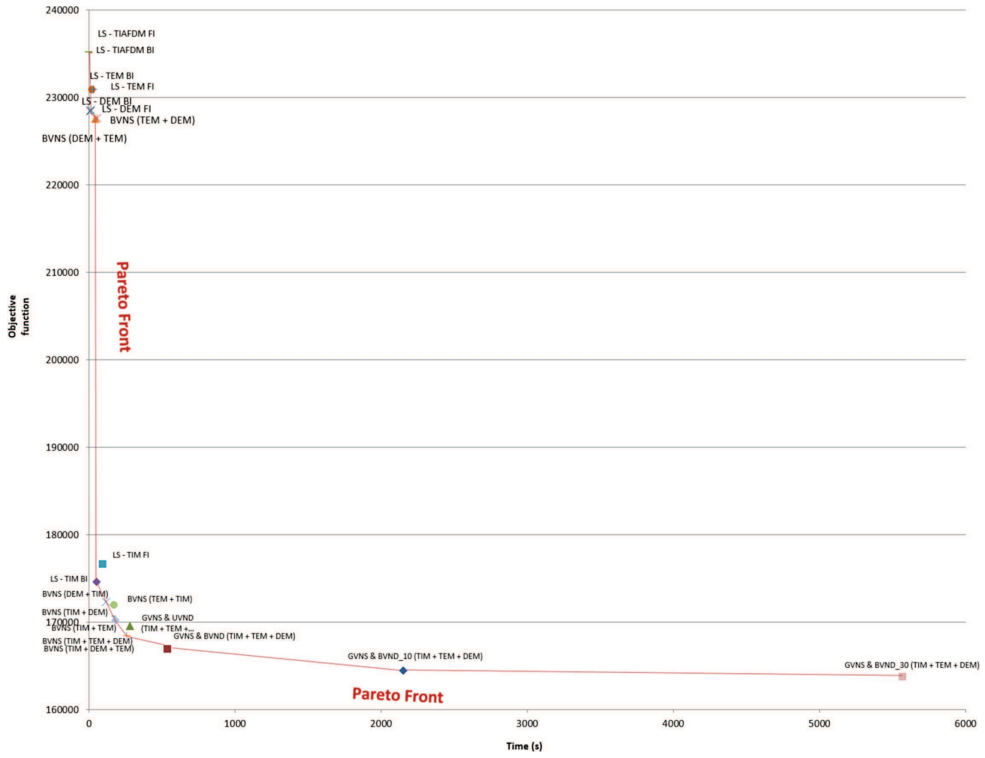

FIGURE 7. Metaheuristics distribution according to their performance with Pareto front drawing.

TABLE 9. Average gap of the objective value with CPLEX on the small instances (from 10×3 to 18×4) expressed in percentage.

	TIM		DEM		TEM		TIAFDM	
LS	BI	FI	BI	FI	BI	FI	BI	FI
Gap (%)	2.8	4.6	31	31	31.2	31.3	33.3	33.3
BVNS	TIM-DEM		DEM-TIM		TIM-TEM		TEM-TIM	
Gap (%)	1.4		1.9		1.4		2.3	
BVNS	DEM-TEM		TEM-DEM		BVNS_TIMDEMTEM		BVNS_TIMTEMDEM	
Gap (%)	30.7		30.7		1		0.86	
GVNS	GVNS+B-VND		GVNS+U-VND		GVNS+B-VND_10		GVNS+B-VND_30	
Gap (%)	0.9		1.35		0.4		0.3	

(3) The best VNS variants enable to find at least one time (over the 35 runs) the optimal solution for almost all the instances (see lignes "min" in the table of results of Appendix A).
(4) For each group of instances, GVNS+B-VND_10 and GVNS+B-VND_30 always find at least one time the optimal solution. For some group, they enable to find the optimal solution (hence their gap of 0.4 and 0.3 respectively).

In order to reveal significant differences between CPLEX and the more efficient variants of this study, we apply Friedman test (see Tabs. 10 and 12). We apply this test only to the small instances to allow comparison with

TABLE 10. Results of the Friedman test for solution quality and runtime of CPLEX and VNS algorithms with three operators (with $k = 4$, $b = 8$ and $\alpha = 0.01$).

	CPLEX		BVNS_ TIMTEMDEM		GVNS+ B-VND		GVNS+ U-VND	
	$R(X_{b_1})$	$R(X_{b_1})^2$	$R(X_{b_2})$	$R(X_{b_2})^2$	$R(X_{b_3})$	$R(X_{b_3})^2$	$R(X_{b_4})$	$R(X_{b_4})^2$
Solution quality								
Average rank	1		3.13		2.38		3.5	
Summation	8	8	25	83	19	47	28	102
Runtime								
Average rank	4		1		3		2	
Summation	32	128	8	8	24	72	16	32
			Solution quality				Runtime	
T_2			19.05					
$F_{0.99,3,21}$					4.87			

TABLE 11. Results of the post-hoc test for solution quality and runtime of CPLEX and VNS algorithms with three operators (with $\alpha = 0.05$ and 21 degrees of liberty).

(a) Solution quality

	BVNS_TIMTEMDEM	GVNS+B-VND	GVNS+U-VND
CPLEX	17 (–)	11 (–)	20 (–)
Critical value		10.64	

(b) Runtime

	BVNS_TIMTEMDEM	GVNS+B-VND	GVNS+U-VND
CPLEX	24	8	16
Critical value		10.64	

TABLE 12. Results of the Friedman test for solution quality of CPLEX and VNS algorithms with $n_{\max} \in \{1; 10; 30\}$ (with $k = 4$, $b = 8$ and $\alpha = 0.01$).

	CPLEX		GVNS+ B-VND		GVNS+ B-VND_10		GVNS+ B-VND_30	
	$R(X_{b_1})$	$R(X_{b_1})^2$	$R(X_{b_2})$	$R(X_{b_2})^2$	$R(X_{b_3})$	$R(X_{b_3})^2$	$R(X_{b_4})$	$R(X_{b_4})^2$
Average Rank	1.25		4		2.88		1.88	
Summation	10	13.5	32	128	23	67	15	28.5
$F_{0.99,3,21}$					4.87			

the optimal solutions provided by CPLEX. Both in term of solution quality and runtime, the null hypothesis is rejected. The post-hoc analysis (see Tabs. 11 and 13) shows that CPLEX is the slowest, but in term of solution quality, it outperforms the metaheuristics excluding GVNS+B-VND_30. Indeed, CPLEX and GVNS+B-VND_30 have the same performance.

For the instances 18×6 and 20×6, CPLEX does not find the optimal solution in all instances, it provides only lower bounds. For the instance 18×6, CPLEX outperforms our algorithms, but these last ones are able to find at least one solution with a better quality. For the instances 20×6, the three GVNS+B-VND variants outperform CPLEX and these algorithms enable to find at least one time an even better solution.

TABLE 13. Results of the post-hoc test for solution quality of CPLEX and VNS algorithms with $n_{\max} \in \{1; 10; 30\}$ (with $\alpha = 0.05$ and 21 degrees of liberty).

	GVNS+B-VND	GVNS+B-VND_10	GVNS+B-VND_30
CPLEX	22 (–)	13 (–)	5(–)
Critical value		10.64	

For large instances (from 20×8), CPLEX is not able to provide optimal solution in less than two hours. Whereas, our best algorithms provide near optimal solutions within a computational time between 0.02 s and 40 min.

This final analysis highlights the efficiency of the best VNS: BVNS_TIMTEMDEM and GVNS+B-VND variants. In particular, GVNS+B-VND_30 is the only one to have the same performance as CPLEX in term of solution quality. We may conclude that to solve the cross-dock problem at best with VNS, it is preferable to perform GVNS combined with a B-VND using Best Improvement procedure and with TIM-TEM-DEM as ordered list of neighborhood operators. If this algorithm stops whenever the solution cannot be improved through at least 30 iterations, GVNS achieves high performance.

7. Conclusion and perspective

To operate cross-dock successfully, an efficient assignment of trucks to docks is one of the key decisions. In this paper, we sought to find an optimal assignment for each truck within its time window to increase the efficiency of the cross-dock and to minimize the total cost of transferring shipments within the cross-dock and the penalty cost of non-delivering goods.

First, we proposed an amended ILP model based on one proposed by Miao *et al.* [21]. To get optimal solutions the CPLEX software were used. Since the problem is NP-hard, it was unlikely to optimally solve the problem in a reasonable time.

Hence, we proposed a VNS metaheuristic to solve this problem approximately. Because different VNS variants can be obtained depending on different configurations and setting parameters, the main purpose of this paper is to deliver a wide strucutred experimental study and a sensitivity analysis to compare these variants. The aim was to highlight the most effective VNS configuration for solving the cross-dock problem at best. First of all, we provided four neighborhood operators (TIM, TEM, DEM, TIAFDM) and compared them within a local search so as to illustrate their characteristics and their efficiency. This step allowed also to compare two search strategies: BI *versus* FI. Then, we tested the influence on the solution quality and runtime of the order of operators within BVNS using two then three operators. Moreover, we compared three VNS variants: BVNS that uses a local search *versus* GVNS that uses a B-VND *versus* GVNS that uses U-VND. Finally, we evaluated the influence of the stopping criterion within VNS.

This empirical study provided many answers. First, the behavior of the operator TIM illustrated that it is a typical intensification operator while DEM and TEM perform as a diversification operator; TIM-TEM-DEM in this order combined with the best improvement strategy is most successful for VNS; GVNS using B-VND significantly outperforms BVNS and GVNS using U-VND; increasing the value of $n_{\max}$ for the stopping criterion improves solution quality; GVNS that uses B-VND with $n_{\max} = 30$ has the same performance as CPLEX while consuming much less time.

A perspective to this work would be to include other VNS settings in the empirical study, like the initial solution or other VNS variants. An hybridisation schema could be used to solve the cross-dock problem, for example matheuristics based on integrating metaheuristics and exact methods. In addition, we can evolve this problem, for example by taking into consideration the capacity of trucks or considering soft time windows for trucks which have been discarded in the current version of this model.

Appendix A. Table of results

LS – operator with strategy BI or FI.

VNS variant (first operator + ... + last operator) with strategy BI.

For CPLEX results, *Obj* is the value of the objective function provided in $TIME$ second, and LB is the lower bound given.

For metaheuristics results, *Obj* is the average value of the objective function, $Time$ is the average runtime in second, σ is the standard deviation and *min* the value of the smallest objective function. *Gap* is the difference between the objective function given by the metaheuristic *Obj* and the lower bound LB given by CPLEX, expressed as a percentage.

The columns $Total$ and *Average* show respectively the sum and the average results over entire set of instances.

In each table and for each instances group, the best average objective function provided by a metaheuristic is in bold characters, as well as the best average runtime consumed.

TABLE A.1. Results of Local Search on instances with small and medium sizes.

			10×3	12×4	12×6	14×4	14×6	16×4	16×6	18×4	18×6	20×6
CPLEX		Obj	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5841.400	9912.800	6062.000	10 784.200
		LB	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5707.994	9912.800	5249.283	713.076
		Time (s)	1.048	31.338	849.402	192.656	2585.738	363.602	5548.786	564.646	6610.098	7058.114
LS DEM	BI	Obj	8207.600	10 126.800	4780.600	8536.800	6559.000	10 406.400	8346.600	11 768.800	10 034.200	12 886.800
		Time (s)	0.002	0.006	0.028	0.007	0.037	0.008	0.036	0.009	0.047	0.067
		Gap (%)	8.026	26.269	59.545	21.306	24.521	42.980	46.227	18.723	91.154	–
	FI	Obj	8207.600	10 130.000	4777.400	8539.000	6563.000	10 407.200	8368.200	11 773.400	10 040.800	12 891.600
		Time (s)	0.001	0.007	0.039	0.008	0.055	0.008	0.063	0.012	0.069	0.101
		Gap (%)	8.026	26.309	59.438	21.337	24.597	42.991	46.605	18.770	91.279	–
LS TEM	BI	Obj	8212.200	10 159.200	4785.400	8528.800	6557.800	10 415.400	8350.400	11 927.400	10 146.200	13 194.400
		Time (s)	0.002	0.008	0.048	0.016	0.055	0.017	0.096	0.035	0.115	0.119
		Gap (%)	8.087	26.673	59.705	21.192	24.498	43.104	46.293	20.323	93.287	–
	FI	Obj	8212.200	10 159.200	4784.200	8528.800	6564.200	10 425.800	8351.000	11 932.600	10 158.400	13 198.200
		Time (s)	0.002	0.008	0.073	0.018	0.074	0.017	0.170	0.029	0.133	0.174
		Gap (%)	8.087	26.673	59.665	21.192	24.619	43.247	46.304	20.376	93.520	–
LS TIM	BI	Obj	**7597.800**	**8068.600**	**3171.000**	7064.400	**5363.000**	**7393.200**	**6220.200**	**10 214.600**	**6545.200**	11 030.600
		Time (s)	0.003	0.016	0.053	0.023	0.107	0.049	0.111	0.039	0.201	0.262
		Gap (%)	0.000	0.606	5.827	0.384	1.815	1.580	8.973	3.045	24.688	–
	FI	Obj	7711.200	8124.000	3213.400	**7059.800**	5425.400	7462.600	6574.600	10 510.600	6588.600	**10 960.800**
		Time (s)	0.004	0.019	0.086	0.035	0.140	0.066	0.131	0.042	0.325	0.416
		Gap (%)	1.493	1.297	7.242	0.318	3.000	2.534	15.182	6.031	25.514	–
LS TIAFDM	BI	Obj	8300.800	10 171.200	4919.200	8608.000	6715.800	10 447.200	8709.800	11 949.200	10 598.600	13 318.000
		Time (s)	**0.000**	**0.000**	**0.001**	**0.000**	**0.002**	**0.001**	**0.002**	**0.000**	**0.001**	**0.003**
		Gap (%)	9.253	26.823	64.170	22.318	27.497	43.541	52.590	20.543	101.906	–
	FI	Obj	8300.800	10 171.200	4919.200	8607.600	6717.000	10 447.200	8709.200	11 949.200	10 598.600	13 318.000
		Time (s)	**0.000**	**0.000**	0.002	0.001	**0.002**	**0.001**	**0.002**	**0.000**	**0.001**	**0.003**
		Gap (%)	9.253	26.823	64.170	22.312	27.520	43.541	52.579	20.543	101.906	–

TABLE A.2. Results of Local Search on instances with large sizes.

			20 × 8	25 × 6	25 × 8	30 × 6	30 × 8	35 × 8	40 × 8	50 × 10	60 × 10	Total	Average
CPLEX		Obj	3405.000	9722.600	7784.800	13 682.600	8588.600	18 882.800	22 710.000	62 413.400	73 473.000	291 460.400	15 340.021
		LB	–	–	–	–	–	–	–	–	–	59 780.353	5978.035
		Time (s)	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	23 805.428	2380.543
LS DEM	BI	Obj	5250.600	11 932.000	10 910.200	17 123.600	7736.200	18 690.600	21 118.600	20 542.800	23 448.800	228 407.000	12 021.421
		Time (s)	0.180	0.099	0.275	0.116	0.491	0.381	0.592	1.779	2.670	6.831	0.360
		Gap (%)	–	–	–	–	–	–	–	–	–	338.751	37.639
	FI	Obj	5250.000	11 952.800	10 912.800	17 125.200	7758.200	18 689.400	21 122.600	20 569.200	23 496.000	228 574.400	12 030.232
		Time (s)	0.526	0.137	0.481	0.199	0.846	0.630	0.810	2.952	5.371	12.315	0.648
		Gap (%)	–	–	–	–	–	–	–	–	–	339.353	37.706
LS TEM	BI	Obj	5415.000	12 025.400	11 230.200	17 137.800	8121.600	18 816.600	21 204.400	20 665.600	23 921.600	230 815.400	12 148.179
		Time (s)	0.439	0.212	0.457	0.282	0.903	1.034	2.019	5.434	8.218	19.510	1.027
		Gap (%)	–	–	–	–	–	–	–	–	–	343.163	38.129
	FI	Obj	5424.600	12 075.800	11 144.400	17 131.600	8121.600	18 822.000	21 216.800	20 686.800	23 976.800	230 915.000	12 153.421
		Time (s)	0.617	0.359	0.620	0.440	1.211	1.444	3.288	9.577	12.358	30.614	1.611
		Gap (%)	–	–	–	–	–	–	–	–	–	343.683	38.187
LS TIM	BI	Obj	**3633.800**	**9700.400**	**7707.800**	13 322.400	**5785.600**	14 993.400	15 748.400	**14 311.000**	**16 726.800**	**174 598.200**	**9189.379**
		Time (s)	0.541	0.491	0.911	0.933	1.581	2.326	3.536	15.429	25.301	51.913	2.732
		Gap (%)	–	–	–	–	–	–	–	–	–	46.917	5.213
	FI	Obj	3720.800	10 073.600	8141.200	**13 217.800**	5869.600	**14 631.000**	**15 513.800**	14 965.800	16 814.200	176 578.800	9293.621
		Time (s)	0.513	0.603	1.435	1.195	2.257	4.112	6.638	27.302	48.661	93.981	4.946
		Gap (%)	–	–	–	–	–	–	–	–	–	62.610	6.957
LS TIAFDM	BI	Obj	5550.200	12 192.400	11 374.600	17 344.400	8378.800	19 009.000	21 486.400	21 307.600	24 696.800	235 078.000	12 372.526
		Time (s)	0.004	0.004	**0.003**	**0.008**	**0.011**	**0.020**	**0.011**	**0.112**	**0.040**	**0.222**	**0.012**
		Gap (%)	–	–	–	–	–	–	–	–	–	368.641	40.960
	FI	Obj	5550.600	12 303.400	11 371.800	17 344.400	8378.000	19 009.000	21 485.800	21 313.400	24 709.400	235 203.800	12 379.147
		Time (s)	**0.003**	**0.003**	**0.003**	0.010	0.013	0.021	0.012	0.121	0.048	0.247	0.013
		Gap (%)	–	–	–	–	–	–	–	–	–	368.647	40.961

TABLE A.3. Results of BVNS with 2 operators on instances with small and medium sizes.

		10×3	12×4	12×6	14×4	14×6	16×4	16×6	18×4	18×6	20×6
CPLEX	Obj	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5841.400	9912.800	6062.000	10 784.200
	LB	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5707.994	9912.800	5249.283	713.076
	Time (s)	1.048	31.338	849.402	192.656	2585.738	363.602	5548.786	564.646	6610.098	7058.114
BVNS (TEM-TIM)	Obj	7676.034	8241.069	3031.954	7096.737	5419.331	7392.560	5940.811	10 335.897	6560.120	10 978.154
	Time (s)	0.023	0.097	0.351	0.142	0.409	0.207	0.617	0.264	0.818	1.059
	σ	167.118	461.830	66.017	80.244	223.396	187.520	134.523	374.245	617.714	218.630
	min	7597.800	8020.000	3000.400	7038.600	5275.800	7279.400	5847.000	9919.400	6118.600	10 754.200
	Gap (%)	1.030	2.756	1.187	0.843	2.884	1.571	4.079	4.268	24.972	–
BVNS (TIM-TEM)	Obj	**7597.800**	**8090.749**	3025.657	**7058.331**	5341.971	**7333.886**	**5894.046**	10 268.103	6430.457	10 856.834
	Time (s)	0.017	0.079	0.328	0.105	0.402	0.199	0.598	0.226	0.813	1.115
	σ	0.000	183.056	31.950	53.904	107.398	100.491	81.667	390.901	405.178	136.884
	min	7597.800	8020.000	2996.400	7037.400	5274.600	7278.200	5847.600	9922.200	6075.800	10 755.600
	Gap (%)	0.000	0.882	0.976	0.297	1.416	0.765	3.259	3.584	22.502	–
BVNS (DEM-TIM)	Obj	7668.846	8155.411	3035.411	7081.360	5368.503	7373.737	5935.080	10 274.377	**6384.120**	10 975.983
	Time (s)	0.016	0.066	0.252	0.096	0.338	0.142	0.478	0.167	0.652	0.773
	σ	158.714	369.418	76.810	77.230	194.356	185.803	150.720	374.109	462.909	348.424
	min	7597.800	8020.000	2998.200	7040.600	5268.000	7291.800	5851.800	9923.600	6077.400	10 749.200
	Gap (%)	0.935	1.688	1.302	0.625	1.919	1.313	3.978	3.648	21.619	–
BVNS (TIM-DEM)	Obj	7615.183	8104.943	**3018.537**	7067.811	**5323.377**	7356.606	5921.177	**10 161.617**	6403.931	**10 854.834**
	Time (s)	0.017	0.073	0.309	0.091	0.423	0.169	0.568	0.192	0.811	0.991
	σ	43.184	187.237	25.315	56.600	108.869	119.289	131.330	391.203	552.793	107.346
	min	7597.800	8020.000	3000.200	7037.600	5271.200	7287.200	5853.000	9926.000	6080.200	10 758.600
	Gap (%)	0.229	1.059	0.739	0.432	1.063	1.077	3.735	2.510	21.996	–
BVNS (TEM-DEM)	Obj	8206.400	10 125.611	4766.771	8519.709	6540.469	10 378.114	8335.783	11 739.240	10 005.360	12 869.206
	Time (s)	0.009	0.047	0.186	0.071	0.204	0.091	**0.269**	0.154	0.331	0.491
	σ	0.000	0.778	6.228	2.004	5.687	4.999	14.764	5.748	8.826	11.994
	min	8206.400	10 124.600	4755.200	8516.200	6532.800	10 375.000	8314.600	11 733.600	9991.600	12 849.400
	Gap (%)	8.010	26.255	59.083	21.063	24.169	42.592	46.037	18.425	90.604	–
BVNS (DEM-TEM)	Obj	8207.600	10 124.943	4764.606	8520.846	6546.714	10 379.023	8317.269	11 743.349	10 008.709	12 860.389
	Time (s)	**0.007**	**0.036**	**0.180**	**0.064**	**0.177**	**0.079**	0.293	**0.128**	**0.277**	**0.383**
	σ	0.000	1.142	6.298	3.617	5.350	6.097	6.414	6.718	4.940	11.671
	min	8207.600	10 124.000	4754.400	8516.400	6537.200	10 375.000	8309.400	11 733.000	10 000.800	12 849.000
	Gap (%)	8.026	26.246	59.011	21.079	24.287	42.604	45.713	18.467	90.668	–

TABLE A.4. Results of BVNS with 2 operators on instances with large sizes.

		20 × 8	25 × 6	25 × 8	30 × 6	30 × 8	35 × 8	40 × 8	50 × 10	60 × 10	Total	Average
CPLEX	Obj	3405.000	9722.600	7784.800	13 682.600	8588.600	18 882.800	22 710.000	62 413.400	73 473.000	291 460.400	15 340.021
	LB	–	–	–	–	–	–	–	–	–	59780.353	5978.035
	Time (s)	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	23 805.428	2380.543
BVNS (TEM-TIM)	Obj	3559.709	9601.857	7487.474	12 950.840	5580.617	14 759.149	15 587.023	13 753.257	15 983.120	171 935.714	9049.248
	Time (s)	2.483	1.809	4.106	3.060	7.714	9.016	14.930	46.853	78.213	172.173	9.062
	σ	292.568	371.612	654.354	605.239	396.030	541.306	684.256	797.993	650.437	7525.033	396.054
	min	3314.800	9235.600	7010.600	12 395.200	5293.400	14 088.800	14 669.600	12 825.400	14 964.200	164 648.800	8665.726
	Gap (%)	–	–	–	–	–	–	–	–	–	43.590	4.843
BVNS (TIM-TEM)	Obj	**3378.383**	**9579.960**	**7353.897**	**12 741.160**	**5516.880**	**14 616.417**	15 341.686	**13 716.091**	16 058.143	**170 200.451**	**8957.918**
	Time (s)	2.661	1.725	4.558	3.262	7.762	9.248	14.736	52.832	81.867	182.532	9.607
	σ	123.696	329.184	508.291	422.636	265.400	512.039	669.389	807.372	638.765	5768.201	303.590
	min	3300.000	9237.400	7005.000	12 380.600	5292.800	14 079.600	14 454.400	12 834.000	14 951.200	164 340.600	8649.505
	Gap (%)	–	–	–	–	–	–	–	–	–	33.682	3.742
BVNS (DEM-TIM)	Obj	3420.800	9788.474	7691.577	12 957.057	5578.274	14 748.314	15 476.360	14 076.434	16 286.806	172 276.926	9067.207
	Time (s)	1.744	1.289	3.056	2.209	5.204	6.475	9.708	33.493	51.173	117.332	6.175
	σ	219.767	601.735	642.988	655.782	476.907	503.687	841.901	975.295	847.774	8164.328	429.701
	min	3311.000	9239.800	7020.200	12 406.000	5311.000	14 114.200	14 492.400	12 798.800	14 966.200	164 478.000	8656.737
	Gap (%)	–	–	–	–	–	–	–	–	–	37.027	4.114
BVNS (TIM-DEM)	Obj	3391.834	9614.200	7544.400	12 864.874	5519.823	14 666.640	**15 308.931**	13 821.509	16 088.834	170 649.063	8981.530
	Time (s)	2.355	1.528	3.895	2.830	6.911	8.773	13.049	46.773	69.730	159.487	8.394
	σ	88.816	310.201	660.990	556.007	378.408	454.573	730.956	883.849	690.884	6477.849	340.939
	min	3318.000	9252.600	6984.200	12 394.800	5304.000	14 098.000	14 457.400	12 636.400	15 030.600	16 4307.800	8647.779
	Gap (%)	–	–	–	–	–	–	–	–	–	32.840	3.649
BVNS (TEM-DEM)	Obj	5218.211	11 887.160	10 873.543	17 076.783	7676.760	18 632.217	21 046.943	20 392.223	23 337.846	227 628.349	11 980.439
	Time (s)	1.579	**0.616**	1.966	1.026	4.238	3.348	5.187	14.096	19.977	53.887	2.836
	σ	15.898	11.067	21.546	11.952	19.575	23.136	16.874	30.610	35.602	247.288	13.015
	min	5182.600	11 872.600	10 845.800	17 057.800	7637.400	18 587.200	21 014.800	20 311.800	23 262.600	227 172.000	11 956.421
	Gap (%)	–	–	–	–	–	–	–	–	–	336.238	37.360
BVNS (DEM-TEM)	Obj	5211.251	11 898.474	10 874.463	17 074.606	7676.246	18 625.549	21 025.617	20 401.423	23 307.914	227 568.989	11 977.315
	Time (s)	**1.102**	0.622	**1.427**	**0.879**	**2.577**	**2.508**	**4.241**	**11.773**	**14.939**	**41.692**	**2.194**
	σ	19.435	11.731	23.303	6.176	25.167	18.898	23.032	54.198	51.706	285.891	15.047
	min	5179.800	11 883.600	10 834.800	17 062.400	7627.600	18 590.000	20 987.200	20 316.000	23 215.600	22 7103.800	11 952.832
	Gap (%)	–	–	–	–	–	–	–	–	–	336.102	37.345

TABLE A.5. Results of BVNS and GVNS with 3 operators on instances with small and medium sizes

		10×3	12×4	12×6	14×4	14×6	16×4	16×6	18×4	18×6	20×6
CPLEX	Obj	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5841.400	9912.800	6062.000	10 784.200
	LB	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5707.994	9912.800	5249.283	713.076
	Time (s)	1.048	31.338	849.402	192.656	2585.738	363.602	5548.786	564.646	6610.098	7058.114
BVNS (TIM-TEM-DEM)	Obj	7606.897	8035.131	3010.474	7046.023	5294.720	7327.383	5894.983	10 060.777	6275.749	10 808.503
	Time (s)	0.024	0.108	0.491	**0.137**	0.603	0.261	**0.822**	0.368	1.403	1.917
	σ	32.094	39.689	22.515	18.410	38.275	97.274	102.607	322.848	513.464	122.933
	min	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5841.600	9912.800	6060.400	10 744.800
	Gap (%)	0.120	0.189	0.470	0.123	0.519	0.676	3.276	1.493	19.554	–
BVNS (TIM-DEM-TEM)	Obj	7605.331	8086.046	3011.137	7046.234	5308.126	7325.857	5881.869	10 125.640	6255.606	10 818.366
	Time (s)	**0.022**	**0.104**	**0.486**	0.140	**0.542**	**0.257**	0.880	**0.334**	**1.167**	**1.526**
	σ	9.777	163.362	28.498	17.686	94.777	80.180	93.625	377.137	427.957	166.973
	min	7597.800	8020.000	2996.400	7037.400	5269.000	7278.200	5841.400	9913.000	6060.400	10 745.200
	Gap (%)	0.099	0.824	0.492	0.126	0.773	0.655	3.046	2.147	19.171	–
GVNS+U-VND (TIM-TEM-DEM)	Obj	7599.800	8083.897	3014.731	7051.497	5371.406	7323.143	5879.749	10 267.966	6364.406	10 824.594
	Time (s)	0.035	0.155	0.661	0.211	0.733	0.446	1.256	0.530	1.541	1.861
	σ	5.293	167.665	48.458	40.454	176.330	62.268	102.988	473.539	460.938	140.781
	min	7597.800	8020.000	2996.400	7037.400	5267.600	7278.200	5841.200	9912.800	6060.600	10 743.800
	Gap (%)	0.026	0.797	0.612	0.200	1.975	0.617	3.009	3.583	21.243	–
GVNS+B-VND (TIM-TEM-DEM)	Obj	7598.223	8066.491	3002.749	7042.794	5286.606	7324.480	5869.851	10 165.457	6152.029	10 779.286
	Time (s)	0.040	0.188	0.933	0.265	1.016	0.478	1.454	0.602	2.058	2.864
	σ	0.562	144.672	6.649	15.273	32.801	61.576	75.239	408.976	308.164	63.892
	min	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5841.200	9913.200	6060.400	10 741.400
	Gap (%)	0.006	0.580	0.212	0.077	0.365	0.636	2.836	2.549	17.197	–
GVNS+B-VND_10 (TIM-TEM-DEM)	Obj	**7597.800**	8021.937	2997.280	7037.566	5270.743	7280.783	5844.291	9978.069	6066.343	10 750.994
	Time (s)	0.261	1.137	5.360	1.630	5.358	2.700	8.552	3.562	10.711	15.081
	σ	0.000	7.110	1.482	0.659	4.566	10.038	3.802	129.128	14.306	14.625
	min	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5841.200	9912.800	6060.200	10 740.600
	Gap (%)	0.000	0.024	0.029	0.002	0.063	0.035	2.388	0.658	15.565	–
GVNS+B-VND_30 (TIM-TEM-DEM)	Obj	**7597.800**	**8020.000**	**2996.954**	**7037.400**	**5268.097**	**7278.623**	**5842.280**	**9922.777**	**6062.194**	**10 744.257**
	Time (s)	0.761	3.261	14.354	4.500	14.343	7.361	22.121	9.750	28.442	40.215
	σ	0.000	0.000	1.008	0.000	2.104	1.360	1.629	46.685	2.660	4.357
	min	7597.800	8020.000	2996.400	7037.400	5267.400	7278.200	5841.200	9912.800	6060.200	10 740.600
	Gap (%)	0.000	0.000	0.018	0.000	0.013	0.006	2.353	0.101	15.486	–

TABLE A.6. Results of BVNS and GVNS with 3 operators on instances with large sizes.

		20 × 8	25 × 6	25 × 8	30 × 6	30 × 8	35 × 8	40 × 8	50 × 10	60 × 10	Total	Average
CPLEX	Obj	3405.000	9722.600	7784.800	13 682.600	8588.600	18 882.800	22 710.000	62 413.400	73 473.000	291 460.400	15 340.021
	LB	–	–	–	–	–	–	–	–	–	59 780.353	5978.035
	Time (s)	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	≥7200	23 805.428	2380.543
BVNS (TIM-TEM-DEM)	Obj	3334.040	9505.291	7206.909	12 730.143	5429.509	14 481.600	15 131.823	13 504.486	15 706.880	168 391.320	8862.701
	Time (s)	**3.707**	2.530	6.731	**4.208**	12.398	**13.551**	**19.317**	75.774	117.957	262.308	13.806
	σ	36.320	278.272	465.420	429.354	261.763	474.672	619.079	878.594	615.940	5369.523	282.606
	min	3292.400	9222.800	6961.800	12 352.200	5240.600	14 052.400	14 376.400	12 548.000	14 829.600	163 633.000	8612.263
	Gap (%)	–	–	–	–	–	–	–	–	–	26.418	2.935
BVNS (TIM-DEM-TEM)	Obj	3336.520	9526.657	7273.794	12 657.131	5411.349	14 440.783	15 062.514	13 643.543	15 603.480	168 419.983	8864.210
	Time (s)	3.711	**2.428**	**6.233**	4.415	**11.759**	13.774	20.707	**73.116**	**115.362**	**256.963**	**13.524**
	σ	47.485	255.089	528.277	409.339	240.714	436.814	609.785	942.295	570.244	5500.015	289.474
	min	3284.400	9227.400	6959.200	12 353.800	5244.600	14 061.800	14 367.600	12 547.800	14 724.600	163 530.000	8606.842
	Gap (%)	–	–	–	–	–	–	–	–	–	27.332	3.037
GVNS+U-VND (TIM-TEM-DEM)	Obj	3397.897	9574.017	7345.680	12 780.817	5518.509	14 556.737	15 347.857	13 421.503	15 789.349	169 513.554	8921.766
	Time (s)	4.670	3.026	7.297	4.916	13.097	15.053	23.272	78.338	121.127	278.225	14.643
	σ	254.636	243.825	540.231	548.361	492.225	511.812	699.191	864.670	702.635	6536.299	344.016
	min	3283.000	9310.000	6957.800	12 357.200	5234.800	14 057.600	14 409.000	12 531.000	15 008.200	163 904.400	8626.547
	Gap (%)	–	–	–	–	–	–	–	–	–	32.062	3.562
GVNS+B-VND (TIM-TEM-DEM)	Obj	3326.383	9435.074	7063.714	12 532.920	5360.394	14 322.840	15 029.840	13 149.709	15 401.234	166 910.074	8784.741
	Time (s)	6.637	4.689	11.278	8.404	22.777	26.002	42.457	151.065	256.042	539.252	28.382
	σ	66.876	200.436	259.559	281.150	217.096	438.413	491.903	617.590	583.153	4273.980	224.946
	min	3283.600	9220.200	6957.800	12 351.400	5228.400	14 048.800	14 361.400	12 481.200	14 638.000	163 324.200	8596.011
	Gap (%)	–	–	–	–	–	–	–	–	–	24.456	2.717
GVNS+B-VND_10 (TIM-TEM-DEM)	Obj	3294.686	9331.263	6965.149	12 394.143	5266.343	14 128.446	14 669.366	12 696.863	14 873.611	164 465.674	8656.088
	Time (s)	37,755	24,814	56.170	38.809	107.391	117.748	196.946	677.267	838.118	2149.371	113.125
	σ	11.489	145.287	17.218	110.516	94.506	217.332	364.880	407.752	320.824	1875.520	98.712
	min	3282.000	9217.200	6944.400	12 341.400	5218.400	14 034.000	14 342.000	12 427.200	14 605.200	16 3163.800	8587.568
	Gap (%)	–	–	–	–	–	–	–	–	–	18.766	2.085
GVNS+B-VND_30 (TIM-TEM-DEM)	Obj	**3288.554**	**9225.777**	**6956.451**	**12 361.463**	**5240.326**	**14 071.977**	**14 496.869**	**12 585.349**	**14 747.069**	**16 3744.217**	**8618.117**
	Time (s)	97.899	65.476	144.732	98.178	269.490	288.658	480.169	1588.587	2390.189	5568.485	293.078
	σ	7.017	12.058	7.715	31.122	36.617	107.585	232.871	285.557	250.039	1030.382	54.231
	min	3281.200	9217.200	6942.800	12 340.800	5214.600	14 029.400	14 328.400	12 401.800	14 573.600	163 081.800	8583.253
	Gap (%)	–	–	–	–	–	–	–	–	–	17.977	1.997

Acknowledgements. This work is a part of the ELSAT2020 project. The ELSAT2020 project is co-financed by the European Union with the European Regional Development Fund, the French state and the Hauts de France Region Council.

References

[1] G. Alpan, A.-L. Ladier, R. Larbi and B. Penz, Heuristic solutions for transshipment problems in a multiple door cross docking warehouse. *Comput. Ind. Eng.* **61** (2011) 402–408.

[2] G. Alpan, R. Larbi and B. Penz, A bounded dynamic programming approach to schedule operations in a cross docking platform. *Comput. Ind. Eng.* **60** (2011) 385–396.

[3] U.M. Apte and S. Viswanathan, Effective cross docking for improving distribution efficiencies. *Int. J. Logistics* **3** (2000) 291–302.

[4] O. Babic, D. Teodorovic and V. Tošic, Aircraft stand assignment to minimize walking. *J. Trans. Eng.* **110** (1984) 55–66.

[5] A.R. Boloori Arabani, S.M.1T. Fatemi Ghomi and M. Zandieh, Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage. *Expert Syst. App.* **38** (2011) 1964–1979.

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

[7] P. Buijs, I.F.A. Vis and H.J. Carlo, Synchronization in cross-docking networks: a research classification and framework. *Eur. J. Oper. Res.* **239** (2014) 593–608.

[8] F. Chen and C.-Y. Lee, Minimizing the makespan in a two-machine cross-docking flow shop problem. *Eur. J. Oper. Res.* **193** (2009) 59–72. ISSN 0377-2217.

[9] Y. Cohen and B. Keren, Trailer to door assignment in a synchronous cross-dock operation. *Int. J. Logistics Syst. Manag.* **5** (2009) 574–590.

[10] W.J. Conover, The Friedman test. In: Practical Nonparametric Statistics. John Wiley & Sons, New York, NY (1999) 369–373.

[11] H. Ding, A. Lim, B. Rodrigues and Y. Zhu, New heuristics for over-constrained flight to gate assignments. *J. Oper. Res. Soc.* **55** (2004) 760–768.

[12] H. Ding, A. Lim, B. Rodrigues and Y. Zhu, The over-constrained airport gate assignment problem. *Comput. Oper. Res.* **32** (2005) 1867–1880.

[13] R. Dondo and J. Cerdá, The heterogeneous vehicle routing and truck scheduling problem in a multi-door cross-dock system. *Comput. Chem. Eng.* **76** (2015) 42–62.

[14] G.B. Fonseca, T.H. Nogueira and M.G. Ravetti, A hybrid Lagrangian metaheuristic for the cross-docking flow shop scheduling problem. *Eur. J. Oper. Res.* **275** (2019) 139–154.

[15] S. Gelareh, R. N. Monemi, F. Semet and G. Goncalves, A branch-and-cut algorithm for the truck dock assignment problem with operational time constraints. *Eur. J. Oper. Res.* **249** (2016) 1144–1152.

[16] P. Hansen, N. Mladenović, R. Todosijević and S. Hanafi, Variable neighborhood search: basics and variants. *EURO J. Comput. Optim.* **5** (2017) 423–454.

[17] A.-L. Ladier and G. Alpan, Cross-docking operations: current research *versus* industry practice. *Omega* **62** (2016) 145–162.

[18] A. Lim, B. Rodrigues and Y. Zhu, Airport Gate Scheduling with Time Windows. *Artif. Intell. Rev.* **24** (2005) 5–31.

[19] A. Lim, H. Ma, and Z. Miao, Truck dock assignment problem with operational time constraint within crossdocks, edited by M. Ali and R. Dapoigny. In: Advances in Applied Artificial Intelligence, Vol. 4031 of *Lecture Notes in Computer Science*. Springer, Berlin-Heidelberg (2006) 262–271.

[20] A. Lim, H. Ma and Z. Miao, Truck dock assignment problem with time windows and capacity constraint in transshipment network through crossdocks. In: Computational Science and Its Applications – ICCSA 2006, Vol. 3982 of *Lecture Notes in Computer Science*. Springer, Berlin-Heidelberg (2006) 688–697.

[21] Z. Miao, A. Lim and H. Ma, Truck dock assignment problem with operational time constraint within crossdocks. *Eur. J. Oper. Res.* **192** (2009) 105–115.

[22] Z. Miao, S. Cai and D. Xu, Applying an adaptive tabu search algorithm to optimize truck-dock assignment in the crossdock management system. *Expert Syst. App.* **41** (2014) 16–22.

[23] Z. Miao, J. Zhang, Y. Lan and R. Su, A two-stage genetic algorithm for the truck-door assignment problem with limited capacity vehicles and storage area. *J. Syst. Sci. Syst. Eng.* **28** (2019) 285–298.

[24] A. Mjirda, R. Todosijević, S. Hanafi, P. Hansen and N. Mladenović, Sequential variable neighborhood descent variants: an empirical study on the traveling salesman problem. *Int. Trans. Oper. Res.* **24** (2017) 615–633.

[25] N. Mladenović and P. Hansen, Variable neighborhood search. *Comput. Oper. Res.* **24** (1997) 1097–1100.

[26] A. Mohtashami, Scheduling trucks in cross docking systems with temporary storage and repetitive pattern for shipping trucks. *Appl. Soft Comput.* **36** (2015) 468–486.

[27] A. Mohtashami, M. Tavana, F.J. Santos-Arteaga and A. Fallahian-Najafabadi, A novel multi-objective meta-heuristic model for solving cross-docking scheduling problems. *Appl. Soft Comput.* **31** (2015) 30–47.

[28] W. Nassief, I. Contreras and R. As'ad, A mixed-integer programming formulation and Lagrangean relaxation for the cross-dock door assignment problem. *Int. J. Prod. Res.* **54** (2016) 494–508.

[29] W. Nassief, I. Contreras and B. Jaumard, A comparison of formulations and relaxations for cross-dock door assignment problems. *Comput. Oper. Res.* **94** (2018) 76–88.

[30] T. Obata, The quadratic assignment problem: evaluation of exact and heuristic algorithms. (1979).
[31] S. Sandal, Staging approaches to reduce overall cost in a crossdock environment. Ph.D. thesis, University of Missouri–Columbia, CO (2005).
[32] G.D. Taylor and J.S. Noble, Determination of staging needs in a crossdock environment. In: Proceedings of 2004 Industrial Engineering Research Conference (2004).
[33] L.Y. Tsui and C.-H. Chang, An optimal solution to a dock door assignment problem. *Comput. Ind. Eng.* **23** (1992) 283–286.
[34] B. Vahdani and M. Zandieh, Scheduling trucks in cross-docking systems: robust meta-heuristics. *Comput. Ind. Eng.* **58** (2010) 12–24.
[35] J. Van Belle, P. Valckenaers and D. Cattrysse, Cross-docking: State of the art. *Omega* **40** (2012) 827–846.
[36] I.F.A. Vis and K.J. Roodbergen, Positioning of goods in a cross-docking environment. *Comput. Ind. Eng.* **54** (2008) 677–689.
[37] W. Yu and P.J. Egbelu, Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *Eur. J. Oper. Res.* **184** (2008) 377–396.