
M. B. Fakhrzad\textsuperscript{a}, A. Sadri Esfahani\textsuperscript{b}

Department of Industrial Engineering, Faculty of Engineering, Yazd University, Yazd, Iran

\textbf{Paper Info}

\textbf{Paper history:}
Received 19 May 2013
Received in revised form 22 October 2013
Accepted 21 November 2013

\textbf{Keywords:}
Cross-docking Strategy
Vehicle Routing Problem
Time Windows
Tabu Search
Variable Neighborhood Search

\textbf{Abstract}

In cross docking strategy, arrived products are immediately classified, sorted and organized with respect to their destination. Among all the problems related to this strategy, the vehicle routing problem (VRP) is very important and of special attention in modern technology. This paper addresses the particular type of VRP, called VRPCDTW, considering a time limitation for each customer/retailer. This problem is known as NP-hard problem. Two meta-heuristic algorithms based on the Tabu search (TS) algorithm and variable neighborhood search (VNS) are proposed for its solution. These algorithms are designed for real-world cases and can be generalized to the more complex models such as those which deliveries can be specified in a split form. The proposed TS algorithm also offers a candidate list strategy which has no limitation for the number of nodes and vehicles. A computational experiment is performed to verify our presented algorithms. Through computational experiments, it is indicated that the proposed TS algorithm performs better than VNS algorithm in both aspects of the total cost and computation time.

\textbf{1. Introduction}

Over the recent years, many companies confronted with more complicated and various customer demands. Thus, many companies are trying to achieve high level of agility, flexibility and reliability for various demands [1]. In the real world, the production procedure consists of purchasing raw material from suppliers, producing or manufacturing, storing and delivering the final product to customers. These systems that start by suppliers and end by customers are called supply chain systems [2]. In such systems, operations of a single company necessarily make no improvement in customer’s satisfaction, because its operations may have interacting effects or even adverse effects on other companies in the supply chain system [1]. For this reason, nowadays, supply chain management is one of the most attractive issues in operational management. Apte and Viswanathan [3] express that over 30% of goods price is incurred in distribution process. Thus, the efficient solution on inventory control and distribution management is a vital success factor for companies [4]. In addition, distribution and control of the flow of inventory is one of the major concepts in supply chain management [2]. Typically, five distinct distribution strategies are utilized in the supply chain management. First, strategy is direct shipment in which items are directly shipped from suppliers to the retail stores without going through distribution centers. Milk run is the second distribution strategy. A milk run is a route in which a vehicle/truck delivers product from a single supplier to multiple retailers [5]. Third strategy is known as hub and spoke (H&S). H&S network involves a series of nodes (hubs), connected by arcs (spokes) that represent viable transportation alternatives between two nodes [5]. Pool distribution is the forth strategy. Pool distribution is the distribution of orders to numerous destination points within a particular geographic region. In this strategy, instead of shipping direct from origin supplier to retailer, orders are directly shipped to the regional terminals and then shipped to the retail stores [5]. Cross-docking is the fifth distribution strategy that recently has been regarded [6].
1. 1. Cross-docking

Apte and Viswanathan [7] introduced cross-docking as one of the most strategic and technologic innovations in the supply chain management. In this system, cross docks/distribution centers function as inventory coordination points rather than as inventory storage points [6]. In the cross-docking, all deliverable products arrive to the cross dock and are immediately categorized, sorted and organized according to their destinations and customers’ demand. Then, these products are moved to respective destinations, without storing in the cross dock. In other words, in the cross-docking systems, a few stocks are handled in temporary storage [8]. In this strategy, products usually are stored less than 12 hours at the cross dock ([3, 9]), sometime less than an hour [10]. Figure 1 illustrates the flow of material in the cross-docking.

Cross-docking includes two key processes that are depicted in Figure 1. The pickup process is the product/material flow from supplier to the cross dock and the delivery process is the product/material flow from cross dock to customer. The key issues in the pickup and delivery processes are the simultaneous arrival at the cross dock and consolidation, respectively. The purpose of consolidation is to categorize and to sort the products at the cross dock with respect to their destinations. Regarding to the consolidation process, some of vehicles have to unload their entire burden completely and reload another product(s). Also, some of the vehicles have to unload partial of their products and reload another product(s) and some of the vehicles which have extra capacity, are only loaded with the new product(s). The purpose of simultaneous arrival to the cross dock is to reduce the waiting time of vehicles. If vehicles of the pickup fleet could not arrive at the cross-dock simultaneously, then the consolidation process has been postponed. Thereby, it might increase the waiting time and the inventory level at the cross dock [2].

1. 2. Literature Review and Research Motivation

Recently, many studies treated various issues of cross-docking from different viewpoints. These investigations can be divided into two categories: studies that focus on (1) physical aspects of cross docking and (2) operational aspects of cross-docking.

Most of the first-category studies describe the cross-docking concept and its advantages [11-13], physical design of cross dock in cross-docking [3, 14-18], and optimal location of cross dock [19-21]. On the other hand, most of the investigations related to the second-category focused on the trucks scheduling in cross-docking system [8, 22-31]. In addition, over the last 30 years or so, the classical VRP has attended strongly in the literature. The classical VRP involves the service of a set of customers with known demands by a fleet of vehicles from a single distribution center [2].

The main objective of the problem is to design a set of routes starting and ending at the distribution center/depot such that all customers are serviced and the total cost of the set of routes is minimized [32]. Some of the most recent VRP papers, such as those have been published by Lin et al. [33], Cheng and Wang [34], Catay [32], Mirabi et al. [35] and Zachariadis and Kiranoudis [36]. Mosheiov [37] considered the pickup and delivery problems as a VRP problem and proposed the two heuristic algorithms to find a good solution to minimize transportation cost and maximize the efficiency of vehicles. VRP with time windows (VRPTW) can be very helpful encountering with cross-docking problems [1], because one the core issue in such problems is the simultaneous arrival at the cross dock. In general, time window models can be divided to three categories:

- **VRP with hard time window (VRPHTW):** In such models, vehicles have to service customers in the specific time interval and any violation from the service time window HWV is not admissible for customer i, whatsoever.

- **VRP with soft time window (VRPSW):** In such models, vehicles are allowed to service customers before and after the earliest and latest time window bounds, respectively. If any violation occurs from the service time window \([a_i, b_i]\), by introducing appropriate penalties, measure of customer’s non-satisfaction is reflected [38].

- **VRP with hard and soft time window (VRPHSW):** Such models are combination of the two mentioned models. Time window in these models include a hard and a soft interval. Hard interval cannot be violated and soft interval can be violated [38].

Comparatively, a few number of research projects considered both cross-docking and VRP simultaneously. Earlier, Lee et al. [1] proposed a tabu search algorithm (TS) to determine the number of vehicles and the optimal vehicle routing schedule at a cross-dock to minimize the sum of transportation cost and fixed cost of vehicles. Liao et al. [2] developed new TS algorithm and compared its performance with Lee et al.’s TS [1]. Dondo et al. [39] presented a hybrid multi-echelon multi-item distribution network that contained multi-echelon vehicle routing problem with cross-docking in...
supply chain management by minimizing total transportation cost. Hasani-Goodarzi and Tavakkoli-Moghaddam [40] considered a split vehicle routing problem (SVRP) with capacity constraint for multi-product cross-docks. Mousavi and Tavakkoli-Moghaddam [41] presented a two-stage mixed-integer programming (MIP) model for the location of cross-docking centers and vehicle routing scheduling problems with multiple cross-docking centers due to potential applications in the distribution networks. To the best of our knowledge, there have been a few papers that take both cross-docking and time scheduling with time windows into consideration simultaneously. Ma et al. [42] proposed a new shipment consolidation and transportation problem in cross-docking distribution networks by considering setup cost and time window constraint. Ma et al. [42] considered direct rout (supplier-to-retailer) as well as indirect rout (supplier-to-cross dock and cross dock-to-retailer). Ma et al. [42] introduced a sweep heuristic for the VRPCD that determines pickup/delivery routes and schedules simultaneously with the truck scheduling at the terminal. Dondo and Cerdá [43] assumed that the pickup/delivery nodes should be visited within specific hard time windows. Unlike Ma et al. [42] and Dondo and Cerdá [43], this paper considers the multi-commodity consolidation and soft time windows for each delivery node. In this paper, it is assumed that all deliverable products are moved to respective destination without storing in the cross-dock. It is also assumed that direct shipping from the suppliers to retailers is not allowed. In this paper, the primary motivation is to present two algorithms for the solution of the above-described complex model of cross docking strategy.

This paper is organized as follows: section 2 presents the model assumptions and formulation. Section 3 describes the tabu search and variable neighborhood search algorithms for VRPCDTW and presents the steps of the algorithms. Section 4 compares the performance of the proposed algorithm. Finally, section 5 concludes the paper and presents the future research.

2. MODEL ASSUMPTIONS AND FORMULATION

The problem considered in this study, namely VRPCDTW, is VRP cross-docking with time window. The problem formulation is based on the following assumptions:

- A soft time window is considered for each customer.
- A set of identical vehicles is used to transport goods from supplies to retailers.
- Consolidation process must be accomplished at the cross dock.
- The whole process must be completed in the planning horizon.
- Each supplier or retailer can only picked up or delivered once. Each location (pick up/delivery node) cannot be visited by the same vehicle more than once.
- No intermediate storage in the cross-dock is allowed.
- There are no pre-defined vehicles for some suppliers and retailers.

The main objectives are to determine the number of vehicles and the best routes and schedules to minimize the sum of transportation cost, vehicle operation cost and time window violation cost. The notations (sets, parameters and decision variables) and mathematical formulation are as follows. It should be noted that some of the used notations in the proposed model, are similar to that of Wen et al. [10].

- \( P \): set of nodes in the pickup process
- \( D \): set of nodes in the delivery process
- \( 0 \): cross dock index
- \( n \): number of suppliers/retailers
- \( NV \): number of available vehicles
- \( Q \): capacity of the vehicle(pallet)
- \( A \): the fixed time for loading, unloading and reloading at the cross-dock and each node
- \( B \): the time for loading, unloading and reloading a pallet at the cross-dock and each node
- \( t_{cj} \): transportation cost from node \( i \) to node \( j \)
- \( O_i \): operation cost of vehicle \( V \). It should be noted that all of the vehicles have an identical operation cost.
- \( t_g \): travel time between node \( i \) and \( j \)
- \( P_i \): number of pallets loading in pickup node \( i \)
- \( d_i \): number of pallets unloading in delivery node \( i \)
- \( DT_i \): departure time of vehicle \( V \) from node \( i \)
- \( AT_i \): arrival time of vehicle \( V \) at the cross dock
- \( S_i \): service start time of vehicle \( V \) at node \( i \)
- \( ye_i \): amount of start time earliness of vehicle \( V \) at node \( i \)
- \( yl_i \): amount of start time lateness of vehicle \( V \) at node \( i \)
- \( P_e \): unit penalty cost for earliness
- \( P_l \): unit penalty cost for lateness
Decision variables:

- $x^v_{ij}$: if the vehicle $v$ move from the node $i$ to the node $j$, 1, otherwise, 0 ($i, j \in P$ or $D$)
- $u^v_i$: if the vehicle $v$ unload the goods $i$ at the cross dock, 1, otherwise, 0 ($i \in P$)
- $r^v_i$: if the vehicle $v$ load the goods $i$ at the cross dock, 1, otherwise, 0 ($i \in P$)
- $tu^v$: the time at which vehicle $v$ finishes unloading at the cross dock
- $tr^v$: the time at which vehicle $v$ starts reloading at the cross dock
- $V_i$: the time at which goods $i$ is unloaded at the cross dock

Mathematical model:

\[
\begin{align*}
\text{Min} & \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{v=1}^{NV} tc_{ij} x^v_{ij} + 2 \sum_{i=1}^{n} \sum_{j=1}^{n} (P_i y_{vij} + P_i y_{vji}) \\
\text{s.t.} & \quad \sum_{j=1}^{n} \sum_{i=1}^{n} x^v_{ij} = 1, \forall j \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{n} x^v_{ij} = 1, \forall i \\
& \quad \sum_{j=1}^{n} x^v_{ij} \leq 1, \forall v \\
& \quad \sum_{i=1}^{n} x^v_{ij} \leq 1, \forall v \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{n} x^v_{ji} = 0, \forall p, v \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{n} x^v_{ij} \leq NV, \forall v \\
& \quad \sum_{i=1}^{n} p_i x^v_{ij} \leq Q, \forall v \\
& \quad \sum_{i=1}^{n} p_i x^v_{ji} \leq Q, \forall v \\
& \quad \sum_{i=1}^{n} p_i x^v_{ij} \leq Q, \forall v \\
& \quad \sum_{i=1}^{n} p_i x^v_{ji} \leq Q, \forall v \\
& \quad \sum_{i=1}^{n} p_i x^v_{ij} \leq Q, \forall v
\end{align*}
\]

\[
\begin{align*}
\sum_{i,j \in D} d_i x^v_{ij} & \leq Q, \forall v \\
DT^v_j & = (t_{ij} + DT^v_i + A + p_i) x^v_{ij}, \forall v, j \in P \\
DT^v_j & = (t_{ij} + DT^v_i + A + d_i) x^v_{ij}, \forall v, j \in D \\
\sum_{i=1}^{n} \sum_{j=1}^{n} (A + p_i) x^v_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{n} (A + d_i) x^v_{ij} + \\
\sum_{i=1}^{n} \sum_{j=1}^{n} t^v_i x^v_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{n} t^v_j x^v_{ij} & \leq T, \forall v \\
AT^v & = (DT^v_i + t_{ij}) x^v_{ij}, \forall v, i \in P \\
AT^v & = AT^v, \forall v \neq v' \\
S^v & = M(1 - x^v_{ij}) - DT^v_i - t_{ij} \geq 0, \forall v, i, j \\
LB & \leq S^v & \leq UB, \forall v, i \in D \\
yc & \geq c_i - S^v, \forall i \in D \\
yl & \geq S^v - l, \forall v, i \in D \\
u^v_i & \leq r^v_i \leq 1, \forall v, i \in P \\
u^v_i - r^v_i & \leq \sum_{j=1}^{n} x^v_{ij} - \sum_{j=1}^{n} x^v_{i,j}, \forall v, i \in P \\
tr^v & \geq tu^v, \forall v \\
tr^v & \geq v_j - M(1 - u^v_i), \forall v, i \in P \\
tr^v & \geq v_j - M(1 - r^v_i), \forall v, i \in P \\
x^v_{ij}, u^v_i, r^v_i & \in [0,1], \forall v, i, j \in P \cup D \\
tr^v, tr^v, v_j & \geq 0
\end{align*}
\]
Equilibrium equation is shown in Equation (12). In Equation (13), the sum of the total length of the visit to each node and total transportation time must be less than the planning horizon, \( T \). The arrival time at a cross-dock is represented by Equation (14). The constraint for simultaneous arrival to a cross-dock is given in Equation (15). Equation (16) determines the service start time for each node. The constraint for soft time windows is represented in Equation (17). Equations (18) and (19) determine the amount of earliness and lateness of a start time. This fact that a vehicle, \( V \), should unload or reload product, \( i \), depends on its pickup and delivery routes is expressed by Equation (20). Equation (21) shows the linkage between the pickup and delivery process in the consolidation decisions at the cross dock. Equation (22) indicates that a vehicle cannot start reloading until it finishes unloading. Equations (23) and (24) express the time at which vehicle, \( V \), finishes unloading and starts reloading at a cross dock.

3. META-HEURISTICS FOR THE VRPCDTW

Since VRPCDTW is considered as a NP-hard problem [1], an efficient meta-heuristic method is needed to achieve a good solution in a reasonable amount of time. TS was successfully applied to solve the various types of VRP [1, 2, 44, 45], VNS was also successfully applied to solve the multi depot routing problem [46] and scheduling the trucks in cross-docking systems [26]. In this paper, two meta-heuristics algorithms based on TS and VNS are presented to solve the VRPCDTW. Sections 3.1 and 3.2 are devoted to the TS-based and VNS-based algorithms, respectively.

3. 1. A TS-based Meta-heuristic for the VRPCDTW

TS is an iterative local search algorithm, which cycling back to previously visited solution is prevented by the use of memories. TS was originally developed by Glover [47]. Some of the basic components of TS method are: initial solution, neighborhood structure, stopping criteria, tabu list and aspiration criteria. TS, at each iteration, explores the solution space and tries to make the best possible moves from the current solution \( x \) to the best solution \( \tilde{x} \) in its neighborhood \( N(x) \), even if the move may deteriorate the objective function value. A tabu mechanism is put in place to prevent the process from cycling over a sequence of solutions. TS exploits tabu list to prevent cycling and local optima. Some attributes of the past solutions are registered and any solution possessing these attributes may not be considered. Temporarily are also declared tabu for \( \theta \) iterations (it is called tabu tenure). However, tabu moves can be overridden if the aspiration criterion is satisfied. Here, a TS-based heuristic to solve the VRPCDTW is developed.

3. 1. 1. Initial Solution

Similar to the other local search algorithms, TS is an iterative procedure that starts from an initial solution. This solution is the starting point for subsequence exploration in the solution space. Here, an initial solution scheme for VRPCDTW is introduced.

**Pickup process:**
1) Calculate the minimum number of vehicles.
\[ N_{V_{min}} = \sum p_i / Q \]

2) Sequence vehicles in descending order of the remaining space. In the stage of initialization, all of the vehicles are available at the cross dock. They are empty and sort according to their index. \( S_1, S_2, \ldots, S_{N_{V_{min}}} \).

3) Determine all possible routes that the first vehicle can be moved in the sequence of step 2. Calculate the ratio of transportation cost to the minimum transportation cost between the determined routes. Generate the candidate list of routes that their calculated ratios are less than \( \alpha \). Select a route and its related node randomly from the candidate list.

4) Replicate the steps 2 and 3 until all nodes are assigned. If the remaining capacity of a vehicle is less than the supply of node \( i \), select next vehicle in the sequence of step 2. If no vehicle was found, add one additional vehicle to the set of vehicles.

**Delivery process**
1) Calculate the remaining time for delivery process., \( t_{d} = T - t \_p \) where, \( t \_p \) is the remaining time for delivery process, \( T \) is the time horizon and \( t \_p \) is the completion time of the pickup process.

2) All vehicles travel to the corresponding customer without any consolidation. For example, if the first vehicle is made pickups in the nodes 1, 2 and 4, thus, shipment of this vehicle must be delivered to the customers 1, 2 and 4.

3. 1. 2. Objective Function Evaluation

The proposed TS algorithm is based on that of Cordeau et al. [48], in which the infeasible solutions are allowed during the search.

According to the initial solution scheme, the time horizon and time windows may be violated. In addition, these violations can be occurred during the search process. Thus, a penalized objective function \( f(s) \) is considered to evaluate each solution generated from the neighborhood of the current solution. The presented objective function evaluation procedure is based on that of Wen et al. [10]. It is defined as follows:
where, \( f(s) \) is the objective function in the algorithm iterations, \( c(s) \) is the original objective function defined in (1), TV, HWV and SWV are the violation measures for delivery process remaining time, hard time windows and soft time windows, respectively. If the solution is feasible for these three constraints, TV, HWV and SWV are equal to 0. \( \alpha \) and \( \beta \) are the penalty coefficients for the time horizon and hard time window violation, respectively. In order to satisfy the soft time windows, a big penalty coefficient (\( M \)) is considered for SWV. TV, HWV and SWV are defined as follows:

\[
\begin{align*}
TV & = \sum_{i=1}^{NV} \left( \sum_{j \in D} \left( S_{ij} + A + d_{ij} \cdot t_{j0} \right) \cdot x_{ij}^* - t_{ij} \right) \\
HWV & = \sum_{i=1}^{NV} \left( \sum_{j \in D} \left( (S_{ij} + A + d_{ij} \cdot B) \cdot x_{ij}^* - l_{ij} \right) \right) \\
SWV & = \sum_{i=1}^{NV} \left( \sum_{j \in D} \left( (S_{ij} + A + d_{ij} \cdot B) \cdot x_{ij}^* - UB_{ij} \right) \right)
\end{align*}
\]

where, \((x)^{*} = \max(0, x)\).

It should be noted that the initial solution is designed in such a manner that the vehicles capacity have no violation. However, during the search algorithm, vehicle capacity may be violated. Therefore, a penalty coefficient for the capacity violation measure is considered. According to the above discussion, \( f(s) \) is modified as follows:

\[
f(s) = c(s) + \alpha TV(s) + \beta HWV(s) + M \cdot SWV(s) + \gamma CV(s)
\]

where, \( CV \) and \( \gamma \) are the capacity violation measure and its penalty coefficient, respectively.

### 3.1.3. Neighborhood Structure

Neighborhood structure is the transformation mechanism that applies on the current solution to generate candidate solution. Insertion and exchange are the most simple and famous mechanisms to generate neighboring solution in the heuristic algorithms. Recently, 2-opt and CROSS mechanisms have generated good solutions. In the insertion mechanism, one node \( i \) removes from its original vehicle \( K \) and re-inserts to another vehicle \( K' \). However, in the exchange mechanism, two nodes belonging to the two vehicles are swapped. 2-opt operator is applied in this paper. One vehicle is selected randomly and then two routes whose transportation cost between two nodes is more than the other nodes are found. These two selected routes are exchanged with the two corresponding routes in another randomly selected vehicle. If there are not any corresponding routes in the two randomly selected vehicles, then, the sequence of the two routes is changed in the first selected vehicle.

#### 3.1.4. Candidate List Strategy

For a given solution, \( X \), it is computationally too expensive to explore its whole neighborhoods. Thus, in the proposed algorithm, instead of examining all neighborhoods, \( N(X) \), a candidate list consisting of vehicles which have the most number of nodes is examined. For example, if the number of vehicles is more than a certain number (i.e. 5 vehicles), the known percent of vehicles (i.e. 50%) which have the most number of nodes are examined.

#### 3.1.5. Tabu Status and Tabu List

One of the TS's objectives is to encourage the exploration of parts of the solution space that have not been visited previously [45]. The complete solutions are not kept in the tabu list. The attributive memory is used for the tabu list, and the e-attributes of an accepted move are stored [49]. For \( 2-opt \) operator, the tabu status of the solution is defined by the vertex pair \((i, j)\). These two vertices are stored in the tabu list, and any solution possessing these attributes may not be considered and temporarily declare the tabu for \( \Theta \) iterations. In this paper, two tabu lists are defined for both pickup and delivery processes. Each tabu list is an \( n \times n \) matrix, where element \( TABU_{ij} \) specifies the tabu status of arc \((i, k)\) in \( TABU \). If \( TABU_{ij} \leq 0 \), arc \((i, k)\) is not a tabu; otherwise tabu.

#### 3.1.6. Aspiration Criterion

In the TS algorithm, tabs may prohibit attractive moves, even when there is no danger of cycling. Hence, it is necessary to use algorithmic devices to allow one to revoke tabs. These are called aspiration criterion. In this paper, similar to the almost all of the TS implementation, most commonly aspiration criterion is used. In this aspiration criterion, a tabu move can be overridden if it has less objective value than the best solution found so far.

#### 3.1.7. Stopping Criterion

In this paper, when the maximum iteration bound determined by the user is reached, the search is stopped. The iteration number is calculated by the following formula:

\[
E = (1 + \frac{NV - 1}{F})
\]

where, \( E \) and \( F \) are parameters and \( NV \) is the number of vehicles. Hence, the number of iterations increases with \( NV \).
3. 1. 8. Proposed Algorithm Steps

Before describing the algorithm flow and presenting the flowchart of the algorithm, some parameters are introduced in the following:

\[ n_{cv}: \text{current number of the generated candidate solutions, in the pickup side;} \]

\[ n_{cd}: \text{current number of the generated candidate solutions, in the delivery side;} \]

\[ n_{max}: \text{maximum number of the candidate moves;} \]

\[ A_p: \text{set of candidate solutions in the pickup process;} \]

\[ A_d: \text{set of candidate solutions in the delivery process;} \]

\[ count: \text{counter of iterations;} \]

\[ NUIN: \text{current number of iterations without improvement;} \]

\[ iteration_{max}: \text{maximum number of algorithm iterations;} \]

\[ NUI_{max}: \text{maximum number of iterations allowed without improvement;} \]

\[ x^*: \text{the best-found solution;} \]

\[ \eta: \text{adjusting coefficient for the penalty coefficients;} \]

**Step 1 Initialization**

1.1) Generate an initial solution \((X)\) based on section 3.1.1.

1.2) Set the algorithm parameters, \(\alpha, \beta, \gamma, \eta, NUIN_{max}, \) tabu tenure(\(\theta\)), \(TABU(i, k) = \phi, E, F,\) and \(count = 1, NUI = 0\). Calculate \(iteration_{max}\) based on (32).

1.3) Set: \(x^* \leftarrow x\) and \(f(x^* ) \leftarrow f(x).\)

**Step 2 Generate the admissible solution in the pickup side**

Set \(n_{vp} = 1\) and \(A_p = \phi\).

2.1) If \(n_{vp} \geq n_{max}\), go to step 3.

2.2) Based on section 3.1.4, determine the vehicle candidate list for generating the neighborhoods. Apply a predetermined neighbor-generating method described in section 3.3. A solution \(x_{vp}\) is generated from the neighborhood \(N(x)\) of \(X\) and added to the candidate solution. Set, \(A_p \leftarrow A_p \cup x_{vp}\), \(n_{vp} \leftarrow n_{vp} + 1\) and go to step 2.

**Step 3 Select the best move in the pickup side amongst \(A_p\)**

Evaluate all the solutions in \(A_p\) according to section 3.2. Put all non-tabu solution in \(N(x)\), set \(i = 1, x_{best}^p = x_{initial}, \) \(f(x_{best}^p) = f(x_{initial}).\) Put all tabu solutions in \(N^*(x)\).

3.1) If \(i > n_{max}\), go to step 4.

3.2) If \(x_i \in N(x) \cup N^*(x)\) and \(f(x_i) \leq f(x_{best}^p)\), set: \(x_{best}^p \leftarrow x_i, i \leftarrow i + 1\) and go to step 3.1.

**Step 4 Generate the admissible solution in the delivery side**

Set \(n_{vd} = 1\) and \(A_d = \phi\).

4.1) If \(n_{vd} \geq n_{max}\), go to step 5.

4.2) Based on section 3.1.4, determine the vehicle candidate list from the solution obtained in step 3, for generating the neighborhoods. Apply exchange operator according to section 3.3. A solution \(x_{vd}\) is generated from the neighborhood \(N(x)\) of \(X\) and added to the candidate solution. Set: \(A_d \leftarrow A_d \cup x_{vd}\), \(n_{vd} \leftarrow n_{vd} + 1\) and go to step 4.1.

**Step 5 Select the best move in the delivery side amongst \(A_d\)**

Evaluate all solutions in \(A_d\) according to section 3.1.2. Put all non-tabu solution in \(N(x)\), set \(i = 1, x_{best} = x_{best}^p, f(x_{best}) = f(x_{best}^p).\) Put all tabu solutions in \(N^*(x)\).

5.1) If \(i > n_{max}\), go to step 6.

5.2) If \(x_i \in N(x) \cup N^*(x)\) and \(f(x_i) \leq f(x_{best})\), set: \(x_{best} \leftarrow x_i, i \leftarrow i + 1\) and go to step 5.1.

**Step 6 Update the statistical information**

If \(f(x_i) \geq f(x_{best})\), set \(NUI \leftarrow NUIN + 1\) and go to step 1. Otherwise: set \(x^* \leftarrow x_{best}\) and \(f(x^*) \leftarrow f(x_{best}).\)

**Step 7 Update the memory structure**

Update the tabu list according to \(x^*\). Set \(TABU(i, k) \leftarrow TABU(i, k) + 1, j = 1, 2\) where the indices 1 and 2 demonstrate the pickup and delivery processes, respectively.

**Step 8 Update the penalty coefficients**

If the current solution is feasible with respect to each item of the objective functions \((TV, HWV, SWV \text{ and } CV)\), the value of the corresponding penalty coefficient is divided by \(1 + \eta\); otherwise, it is multiplied by \(1 + \eta\). Set \(count \leftarrow count + 1\).

**Step 9 Stopping**

If \(count < iteration_{max}\) and \(NUI < NUIN_{max}\), go to step 2; otherwise show the best \(x\) solution.
The flowchart of the proposed algorithm is given in Figure 2. In addition, the pseudo code of the TS algorithm is given below.

1. **Begin**
2. Generate an initial solution.
3. Set the initial solution as the current and also the best solution.
4. Generate an empty list for keeping the frequency of the solution changes in the pickup side.
5. Generate an empty list for keeping the frequency of the solution changes in the delivery side.
6. While stopping criterion is not met do
7. For $n_p = 1$ to $n_{max} = 1$
   - Search the neighborhood of the current solution in the pickup side.
   - Then update the tabu list.
   - If $\text{current solution objective function} < \text{best solution objective function}$
     - best solution ← current solution
9. Endif
10. Endfor
11. current solution ← best solution
12. For $n_d = 1$ to $n_{max} = 1$
   - Search the neighborhood of the current solution in the delivery side.
   - Then, update the tabu list.
   - If $\text{current solution objective function} < \text{best solution objective function}$
     - best solution ← current solution
15. Endif
16. Endfor
17. Endwhile
18. End

3. 2. A VNS-Based Meta-heuristic for the VRPCDTW

VNS is a new meta-heuristic for solving the combinatorial and global optimization problems that proposes systematic changes of the neighborhood structure during the search. VNS originally proposed by Hansen and Mladenovic [50]. Basic idea of the VNS is a systematic changing of neighborhood both within a descent phase to find a local optimum and within a perturbation phase to get out of the corresponding valley [51]. VNS explores close and then increasingly far neighborhoods of the best known solution in a probabilistic way. In other word, VNS applies a local search procedure repeatedly to get from neighboring solutions to local optima [51]. Because of relying on very few parameters, such as stopping criterion and number of neighborhoods, VNS is very easy to implement. A very comprehensive study about VNS can be found in Hansen and Mladenovic [50, 52]. Here, a VNS-based heuristic to solve the VRPCDTW is developed.

3. 2. 1. Initial Solution

Similar to the other meta-heuristic algorithm, an initial feasible solution is necessary to start VNS procedure. In this paper, initial solution scheme described in TS algorithm is applied, as initial solution.

3. 2. 2. Neighborhood Structures

In almost all of the meta-heuristic algorithms, one or more neighborhood structures is utilized as a means of defining admissible moves to transition from one solution to another solution. Because of the good performance, $\text{opt} - N_{\text{opt}}$ operator is applied in this paper.

One vehicle is randomly selected and then two routes whose transportation cost between two nodes is more than the other nodes are found. These two selected routes are exchanged with the two corresponding routes in another randomly selected vehicle. If there are not any corresponding routes in the two randomly selected vehicles, then, the sequence of the two routes is changed in the first selected vehicle.

3. 2. 3. Shake Procedure

The shake procedure generates a $x'$ at random from neighborhood of $x$, i.e. $x' \in N(x)$.

3. 2. 4. Stopping Criterion

In this paper, the stopping criterion is determined by the maximum number of iterations between two improvements. In this paper, the iteration number is calculated by formula (32).

3. 2. 5. Proposed Algorithm Procedure

Step 1 Initialization
1.1) Generate an initial solution ($x$) based on 3.1.1.
1.2) Determine the neighborhood structure. In this paper, $N_{\text{opt}}(.)$.
1.3) Set the algorithm parameters, $\alpha, \beta, \gamma, E, F, \text{count}=1$. Calculate $\text{iteration}_{\text{max}}$ based on formula (32).
1.4) set: $x^* \leftarrow x$ and $f(x') \leftarrow f(x)$.
Step 2 shake and local search in pickup and delivery sides
2.1) Shake routine in the pickup side. Find a random solution \( x' \in N_{2-opt}(x) \).
2.2) Local search in the pickup side. Apply neighborhood structure on \( x' \) to find a solution \( x_{pickup}^* \).
2.3) Shake routine in the delivery side. Find a random solution \( x^* \in N_{2-opt}(x) \).
2.4) Local search in the delivery side. Apply neighborhood structure on \( x^* \) to find a solution \( x_{delivery}^* \).
2.5) If \( f(x_{pickup}) < f(x') \) then \( x' \leftarrow x_{pickup}^* \). If \( f(x_{delivery}) < f(x') \) then \( x^* \leftarrow x_{delivery}^* \). Set: \( count \leftarrow count + 1 \).

Step 3 stopping
If \( count < \text{iteration}_{\text{max}} \) then \( x \leftarrow x' \) and go to 2.1, else show \( x^* \).
The pseudo code of the VNS algorithm is as follows:
1. Begin
2. Generate an initial solution (\( x \)).
3. Set the initial solution as the best solution (\( x^* \)).
4. While stopping criterion is not met do
5. Repeat the following steps
6. \( x \leftarrow x^* \)
7. Shake routine in the pickup side. Find a random solution \( x' \in N_{2-opt}(x) \).
8. Local search in the pickup side. Apply neighborhood structure on \( x' \) to find a solution \( x_{pickup}^* \).
9. Shake routine in the delivery side. Find a random solution \( x^* \in N_{2-opt}(x) \).
10. Local search in the delivery side. Apply neighborhood structure on \( x^* \) to find a solution \( x_{delivery}^* \).
11. If \( f(x_{pickup}) < f(x') \) then
12. \( x^* \leftarrow x_{pickup}^* \)
13. Break
14. If \( f(x_{delivery}) < f(x') \) then

15. \( x^* \leftarrow x_{\text{delivery}} \)
16. Break
17. Endif
18. Endif
19. Endwhile
End

4. COMPUTATIONAL EXPERIMENTS

The aim of this section is to compare the performance of the two proposed algorithms described in section 3. It should be noted that, to verify and validate the mathematical model, several small size test problems were solved by the Lingo 8 software and optimal results were obtained. Optimal results of the four instances are presented in Table 2.

To evaluate the performance of the proposed algorithms, a computational study is carried out. For this purpose, the proposed algorithms are coded by using Visual Basic programming language. In the implemented programs, route of each vehicle is determined as well as respective costs, such as transportation cost, operation cost and earliness and lateness cost. The performances of the proposed algorithms are verified by comparing the obtained results from TS and VNS with the lower bound solution. The lower bound solution is developed by relaxing the constraints (17)-(19) that related to the time windows. To have a fair comparison both algorithms were executed on a special laptop. Due to lack of benchmark problems in the literature, randomly generated problems are considered. The approach of this paper to randomly design of the time windows is similar to Xiangyong et al. [45] approach as follows: The hard time window is specified by \( c_{ij} \), i.e., \([e_i, l_i] = [c_{ei} - \frac{1}{2} tw, c_{ei} + \frac{1}{2} tw] \). \( c_{ei} \) is an integer randomly generated from interval \([0 + t_{oi}, t_{oi} + (A + d_iB)]\) for each node \( i \in D \), where \( t_{oi} = t_{o0} \). Time width \( tw \) is an integer uniformly generated from the range \([10, 30]\). The soft time window is specified by \([LB_{ij}, UB_{ij}] = [e_i - \frac{1}{2} tw, l_i + \frac{1}{2} tw] \). Following Lee et al. [1] other parameter values are as follows: capacity of the vehicle (pallet) \( Q = 70 \), travel time between node \( i \) and \( j \), \( t_{ij} = \text{uniform}(20, 200) \), transportation cost from node \( i \) to node \( j \), \( t_{cj} = \text{uniform}(48, 480) \), number of pallets loading in pickup node \( i \) and number of pallets unloading in delivery node \( j \), \( p_i, d_j = \text{uniform}(5, 50) \). Also, operation cost of vehicle \( v \), \( O \), unit penalty cost for earliness \( P_e \), unit penalty cost for lateness \( P_l \), the fixed time for loading, unloading and reloading at the cross-dock and each node \( A_i \), and the time for loading, unloading and reloading a pallet at the cross-dock and each node \( B_i \), are specified by 100, 5, 5, 10 min and 1 min, respectively. In this paper, the approach of the parameters setting is similar to response surface methodology (RSM). For this purpose, critical factors of the TS algorithm that are statistically significant in aspects of performance and CPU time have been identified. Critical factors of the TS algorithm based on Vahdani and Zandieh [26] are maximum iteration (\( \text{iteration}_{\text{max}} \)), maximum number of iterations allowed without improvement (\( \text{NUI}_{\text{max}} \)), maximum number of the candidate moves (\( n_{max} \)) and tabu tenure (\( \theta \)).

According to the determined neighborhood structure, the critical factor of the VNS algorithm is \( \text{iteration}_{\text{max}} \).

By preliminary experiments, parameters of the proposed algorithms have been tuned, with which the algorithm had a relatively better performance and CPU time. Parameter setting is presented in Table 1.

Total cost comparison results of the two proposed algorithms versus Lingo results for the sample instances are provided in Table 2. The time horizon \( T \) is supposed to be 600 min.

Comparison results show that while using the Lingo software, the solution time is exponentially increased by increasing the number of nodes, whereas the proposed algorithms can solve the problem in much less time. Table 2 also shows that the presented algorithms can generate near optimal solution for the sample problems. Therefore, these algorithms can efficiently solve large size problems in the logical time. Comparative results of total costs and computation times of the TS and VNS algorithms as well as the lower bound solution for randomly generated large size instances are provided in Table 3 and Table 4, respectively. It should be noted that the value of each instance was reported from the average of 10 repetitions. Percentage gap between the total costs of the two presented algorithms and lower bound solution is calculated according to the following formula (I).

\[
\text{Gap} = \frac{\text{total cost of the proposed algorithm} - \text{total cost of the lower bound solution}}{\text{total cost of the lower bound solution}} \times 100
\]
### TABLE 1. Parameter setting

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Value for TS</th>
<th>Value for VNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>penalty coefficients for the time horizon violation</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>$\beta$</td>
<td>penalty coefficients for hard time windows violation</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>penalty coefficient for capacity violation</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>$\eta$</td>
<td>adjusting coefficient for penalty coefficients</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>$E$</td>
<td>parameter for $\text{iteration}_{\text{max}}$ based on formula (32)</td>
<td>60</td>
<td>80</td>
</tr>
<tr>
<td>$F$</td>
<td>parameter for $\text{iteration}_{\text{max}}$ based on formula (32)</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$NUL_{\text{max}}$</td>
<td>maximum number of iterations allowed without improvement</td>
<td>$3n$</td>
<td>-</td>
</tr>
<tr>
<td>$n_{\text{max}}$</td>
<td>maximum number of the candidate moves</td>
<td>50</td>
<td>-</td>
</tr>
<tr>
<td>$\theta$</td>
<td>tabu tenure</td>
<td>$\text{iteration}_{\text{max}}/2$</td>
<td>-</td>
</tr>
</tbody>
</table>

### TABLE 2. Total cost comparison results of the proposed algorithms versus Lingo

<table>
<thead>
<tr>
<th>Problem</th>
<th>No. of available vehicles</th>
<th>No. of nodes in each sides (pickup and delivery)</th>
<th>TS result</th>
<th>VNS result</th>
<th>Lingo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total cost</td>
<td>cpu time (s)</td>
<td>Total cost</td>
</tr>
<tr>
<td>VRPCDTW 1</td>
<td>5</td>
<td>5</td>
<td>2583.3</td>
<td>45</td>
<td>2590</td>
</tr>
<tr>
<td>VRPCDTW 2</td>
<td>5</td>
<td>6</td>
<td>2936.75</td>
<td>61</td>
<td>3020</td>
</tr>
<tr>
<td>VRPCDTW 3</td>
<td>5</td>
<td>7</td>
<td>3145.86</td>
<td>65</td>
<td>3266.9</td>
</tr>
<tr>
<td>VRPCDTW 4</td>
<td>5</td>
<td>8</td>
<td>4257.9</td>
<td>150</td>
<td>4320</td>
</tr>
</tbody>
</table>

### TABLE 3. Total cost obtained by TS algorithm for the large size problems ($T=600$ min)

<table>
<thead>
<tr>
<th>Problem</th>
<th>No. of available vehicles</th>
<th>No. of nodes in each sides (pickup and delivery)</th>
<th>Lower bound solution</th>
<th>TS result</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total cost</td>
<td>Total cost</td>
<td>cpu time (s)</td>
</tr>
<tr>
<td>VRPCDTW 5</td>
<td>10</td>
<td>15</td>
<td>7277</td>
<td>8364</td>
<td>100</td>
</tr>
<tr>
<td>VRPCDTW 6</td>
<td>10</td>
<td>20</td>
<td>8834</td>
<td>10036</td>
<td>210</td>
</tr>
<tr>
<td>VRPCDTW 7</td>
<td>20</td>
<td>25</td>
<td>12035</td>
<td>13299</td>
<td>275</td>
</tr>
<tr>
<td>VRPCDTW 8</td>
<td>20</td>
<td>30</td>
<td>12480</td>
<td>14437</td>
<td>486</td>
</tr>
<tr>
<td>VRPCDTW 9</td>
<td>30</td>
<td>40</td>
<td>15980</td>
<td>18907</td>
<td>840</td>
</tr>
<tr>
<td>VRPCDTW 10</td>
<td>30</td>
<td>50</td>
<td>19750</td>
<td>25003</td>
<td>1022</td>
</tr>
<tr>
<td>VRPCDTW 11</td>
<td>50</td>
<td>60</td>
<td>36551</td>
<td>40167</td>
<td>2306</td>
</tr>
<tr>
<td>VRPCDTW 12</td>
<td>50</td>
<td>70</td>
<td>49383</td>
<td>60225</td>
<td>3126</td>
</tr>
<tr>
<td>VRPCDTW 13</td>
<td>60</td>
<td>80</td>
<td>65511</td>
<td>78930</td>
<td>3840</td>
</tr>
<tr>
<td>VRPCDTW 14</td>
<td>60</td>
<td>90</td>
<td>89127</td>
<td>100128</td>
<td>4209</td>
</tr>
</tbody>
</table>

*Average gap (%) 16.43*

### TABLE 4. Total cost obtained by VNS algorithm for the large size problems ($T=600$ min)

<table>
<thead>
<tr>
<th>Problem</th>
<th>No. of available vehicles</th>
<th>No. of nodes in each sides (pickup and delivery)</th>
<th>Lower bound solution</th>
<th>TS result</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total cost</td>
<td>Total cost</td>
<td>cpu time (s)</td>
</tr>
<tr>
<td>VRPCDTW 5</td>
<td>10</td>
<td>15</td>
<td>6327</td>
<td>9053</td>
<td>121</td>
</tr>
<tr>
<td>VRPCDTW 6</td>
<td>10</td>
<td>20</td>
<td>7474</td>
<td>10101</td>
<td>234</td>
</tr>
<tr>
<td>VRPCDTW 7</td>
<td>20</td>
<td>25</td>
<td>11229</td>
<td>14573</td>
<td>290</td>
</tr>
<tr>
<td>VRPCDTW 8</td>
<td>20</td>
<td>30</td>
<td>13590</td>
<td>15986</td>
<td>503</td>
</tr>
<tr>
<td>VRPCDTW 9</td>
<td>30</td>
<td>40</td>
<td>15963</td>
<td>19716</td>
<td>886</td>
</tr>
<tr>
<td>VRPCDTW 10</td>
<td>30</td>
<td>50</td>
<td>18768</td>
<td>25353</td>
<td>1120</td>
</tr>
<tr>
<td>VRPCDTW 11</td>
<td>50</td>
<td>60</td>
<td>39496</td>
<td>42931</td>
<td>2583</td>
</tr>
<tr>
<td>VRPCDTW 12</td>
<td>50</td>
<td>70</td>
<td>53282</td>
<td>61243</td>
<td>3220</td>
</tr>
<tr>
<td>VRPCDTW 13</td>
<td>60</td>
<td>80</td>
<td>60859</td>
<td>80074</td>
<td>3920</td>
</tr>
<tr>
<td>VRPCDTW 14</td>
<td>60</td>
<td>90</td>
<td>80800</td>
<td>101005</td>
<td>4602</td>
</tr>
</tbody>
</table>

*Average gap (%) 26.45*
As depicted in Table 3, the average gap between the total costs of the TS solution and lower bound solution is 16.43% that is desirable. In addition, Table 4 shows that the average gap between the total costs of the VNS solution and lower bound solution is 26.45%. As a result, computational experiments indicate that the proposed TS algorithm performs better than VNS algorithm in aspect of the total cost.

5. CONCLUSION AND FUTURE RESEARCH

This paper presented a new model, namely VRPCDTW, integrating cross-docking with VRPTW. Since this problem is categorized as a NP-hard problem, two meta-heuristic algorithms based on the TS and VNS were proposed for its solution. The proposed algorithms were shown to have the adequate flexibility while encountering with the real-world cases. A computational experiment was carried out to compare the performance of the proposed algorithms. Experimental results showed that the proposed algorithms were capable of solving the large size problems in the logical time. Because of presenting a candidate list strategy, computational experiments indicated that the proposed TS algorithm performs better than VNS algorithm in both aspects of the total cost and computation time. Future research can be suggested in a few directions. It is interesting to solve the proposed model by new meta-heuristic algorithms and compare their results with the results of the proposed algorithms. Another extension is to consider the fuzzy time window in both sides of the cross-dock (pickup and delivery sides). Future research can be considered the multiple cross-docks, capacity constraint at the cross-dock, direct shipment and non-identical vehicles.

6. REFERENCES


M.B. Fakhrzad¹, A. Sadri Esfahani²

Department of Industrial Engineering, Faculty of Engineering, Yazd University, Yazd, Iran

Keywords: Cross-docking Strategy Vehicle Routing Problem Time Windows Tabu Search Variable Neighborhood Search

Paper history:
Received 19 May 2013
Received in revised form 22 October 2013
Accepted in 21 November 2013

Abstract: In this paper, the Time Windows Vehicle Routing Problem (VRPTW) in a cross-docking strategy is addressed. The problem is NP-hard, and a hybrid of Tabu Search and Variable Neighborhood Search is used to solve it.