Resolution of the vehicle routing problem with cross-docking

Sanae LARIOUI
[1] ENSATE, University of Abdelmalek Essaadi, Mhannech II, BP 2121 Tetouan, Morocco
Sanae.larioui@gmail.com

Mohamed REGHIOUI
[1] ENSATE, University of Abdelmalek Essaadi, Mhannech II, BP 2121 Tetouan, Morocco
m.reghioui@gmail.com

Abstract

In this paper we address the *VRPCD*, in which a set of homogeneous vehicles are used to transport products from the suppliers to customers via a cross-dock. The products can be consolidated at the cross-dock but cannot be stored for very long as the cross-dock does not have long-term inventory-holding capabilities. The objective of the *VRPCD* is to minimize the total traveled distance while respecting time window constraints of suppliers and customers and a time horizon for the whole transportation operation. Rummaging through all the work of literature on vehicle routing problems with cross-docking, there is no work that considers that customer will receive its requests from several suppliers; this will be the point of innovation of this work. To solve this problem, four algorithms are proposed: A memetic algorithm, a tabu search, an evolutionary local search and an Iterated local search are used to solve the problem. The proposed algorithms are implemented and tested on data sets involving up to 200 nodes (customers and suppliers). The first results show that the memetic algorithm can produce high quality solutions.

Keywords
Cross-docking; Vehicle Routing Problem, Memetic algorithm, tabu search, ILS, ELS

1- Introduction

The problem considered in this paper involves a set of known customer orders or requests, each one characterized by the cargo size, the pickup point and the place where it has to be delivered. These requests are picked up by a fleet of homogeneous vehicles, consolidated at the cross-dock, and immediately delivered to customers by the same set of vehicles, without intermediate storage. During the consolidation, goods are unloaded from the inbound vehicles and reloaded on outbound vehicles. In other words, a vehicle starting from the cross-dock first collects several requests at their pickup points, drives back to the cross dock, and unloads some but not necessarily all orders. Some loads may remain in the truck if the same vehicle will transport them to their destinations. Then, the truck moves to the assigned shipping door, loads some additional requests and goes out again to serve the delivery locations. After completing their tours, delivery vehicles return to the cross dock. The objective is to determine the best pickup and delivery routes as well as the arrival times of pickup/delivery vehicles at the cross-dock so that all nodes are visited within their time windows at minimum total transportation cost, including variable and fixed costs.

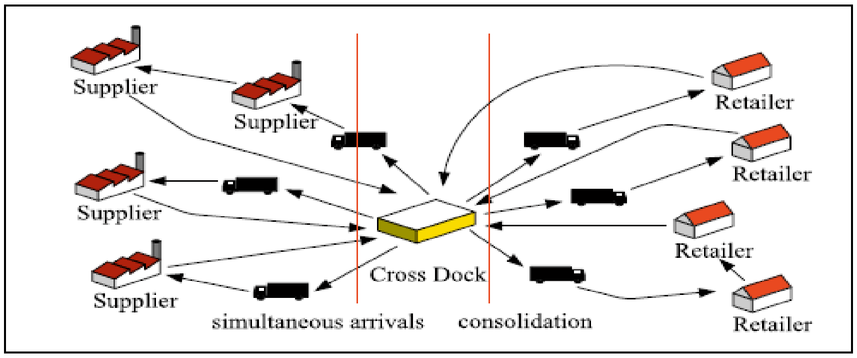

Figure 1.The concept of cross docking in vehicle routing.

Note that in the case without consolidation, the solution of this problem can be found by solving two vehicle routing problems with time windows (one for pickup and one for delivery). But taking into account the consolidation, the pickup and delivery routes are not independent. Trying to minimize the distance of the pickup and delivery routes separately is not sufficient; the exchanges of orders at the cross-dock also have to be taken into account [1], in fact, the problem involves not only vehicle route design, but also a consolidation decision at the cross-dock.

We note that our problem has a difference with that of Wen et al [1], we have added an additional complexity which consists of allowing a customer to order from more than one supplier because we believe that a customer who requests one product from one supplier is a case a little far from what happens in reality. Our case is more explained in the figure below:

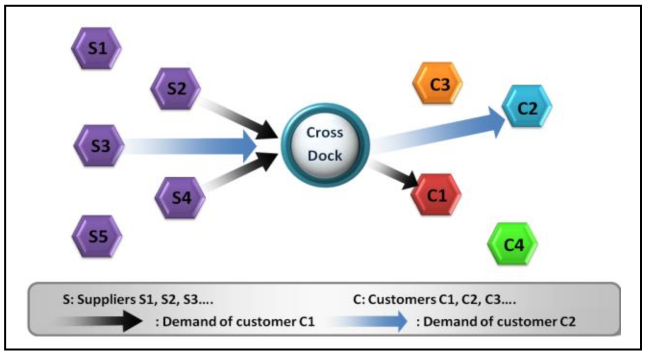

Figure 2.Example of subdivision of client requests

In Figure 2 customer C1 have two types of products ordered from two suppliers S2 and S4.

Figure 3 illustrates the pickup and delivery routes for three vehicles; each vehicle starts and ends their routes at the cross-dock. For example, the first vehicle makes pickups at nodes 1 and 2 and delivers to nodes 1', 2' and 5'. Note that, a delivery vehicle cannot leave the cross dock without loading all requests for all pickup vehicles. For instance, if one client (1') has two suppliers 1 and 3, in this case, the vehicle that will deliver 1' must wait for all vehicles that make pickup to arrive to the cross dock and unload their demands.

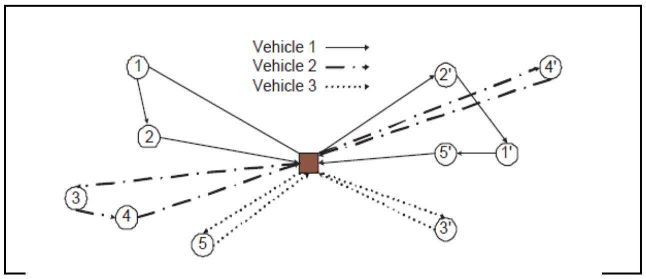

Figure 3.Example of pickup and delivery routes.

Since our VRPCD is based on the VRPTW it is necessary to address this problem which is a generalization of the well-known capacitated vehicle routing problem in which the service of each customer must begin within a specified time window. It is defined on a complete undirected graph G = (V,E) with a node set V = {0, 1, 2, ..., n} and an edge set E. Node 0 represents a depot where a fleet of identical vehicles of capacity W is located. The n other nodes

correspond to the customers. Each customer i has a demand qi and a time window [ei, li], where ei and li are respectively the earliest and the latest service time. Arriving at i earlier 9 than ei is allowed but induces a waiting time ai, while arriving later than li leads 0 to infeasibility. A traversal cost (distance) dij = dji and a traversal time tij = tji are associated with each edge [i, j]. The objective is to build a set of vehicle trips of minimum total cost, such that each trip starts and ends at the depot and services a subset of customers within their time windows. Each customer must be visited by a single trip, i.e., split deliveries are not allowed [12].
As in the Vehicle Routing Problem with Time Windows (VRPTW), each node must be served by exactly one vehicle within its time window, that is why we consider two VRPTWs (one for the pickup problem and one for delivery), the accumulated load of each route must not exceed the vehicle capacity, and the time horizon for the whole transportation operation must be respected. At the cross-dock, the unloading must be completed for each vehicle before reloading starts. Each vehicle can start unloading immediately after it arrives at the cross-dock from its pickup route. The duration of the unloading consists of a fixed time for preparation, and the time needed for unloading products, equal to the time for unloading each pallet multiplied by the number of pallets.
The considered network consists of a set of collecting nodes (Suppliers) P = {1,..., n} and a set of delivery nodes (clients) D = {1,. ..., m}. Each demand Dij is identified by the pair (i, j) with i the pickup node and j the delivery node, it is the sum of the demands from all suppliers for the customer concerned. The cross dock is represented by node 0; which is the departure and the return point of each route of pick up or delivery. Also, the cross dock has a time window which indicates the hours of opening. In addition, the suppliers and the customers must be visited in a specific time window. The objective is to minimize the total cost of transportation respecting the constraints listed above. The complexity of the problem lies in the synchronization required at the cross dock between pickup and delivery routes.
The paper is organized as follows. The mathematical formulation is given in Section 2. Section 3 presents the resolution of the problem. The instances used and the obtained results are discussed in section 4. Finally, the conclusion is presented in section.

2- Mathematical formulation

In this section, an integer linear programming formulation for the VRPCD is proposed; the objective is to minimize the total cost of transportation.
We denote the set of pickup nodes (Suppliers) with P = {1, ..., n) and the set of delivery nodes (customers) with D = {1, ... m}. Each request is identified by the pair (i, j) with i the node of pickup and j the node(s) of delivery. The cross-dock is represented by four nodes and noted by the set O = (o1, o2, o3, o4), the first two nodes represent the start and end points of the pickup.
The last ones are for delivery.
N = is defined PUOUD, all nodes.
The set E indicates all the arcs of the network. They consist of arcs:
{(i,j): i,j ∈ P U { o1; o2},i≠ j}and the arcs {(i,j): i,j ∈ D U { o3; o4}, i≠ j}
K is the set of vehicles
The parameters are:
Cij = the travel time between node i and node j ((i, j) ∈ E);
[ai, bi] = the time window of node i (i ∈ N);
Dij = the amount of demand requested by the customer j to the supplier i (i ∈ P) (j ∈ D);
Q = the vehicle capacity;
A = the fixed time for unloading and reloading at the cross-dock;
B = the time for unloading and reloading a pallet.
The variables are:

$$x_{ij}^{k} = \begin{cases} 1 & \text{if vehicle } k \text{ travels from node } i \text{ to node } j \; ((i;\, j) \in E;\; k \in K) \\ 0 & \text{otherwise} \end{cases}$$

$$u_{ij}^{k} = \begin{cases} 1 & \text{if vehicle } k \text{ unloads request } (i,j) \text{ at the crossdock } (i \in P;\; j \in D, k \in K) \\ 0 & \text{otherwise} \end{cases}$$

$$r_{ij}^{k} = \begin{cases} 1 & \text{if vehicle } k \text{ reloads request } (i,j) \text{at the crossdock } (i \in P; j \in D, k \in K) \\ 0 & \text{otherwise} \end{cases}$$

$$g_k = \begin{cases} 1 \text{ if vehicle } k \text{ has to unload at the crossdock}, k \in K \\ 0 \quad \text{otherwise} \end{cases}$$

$$h_k = \begin{cases} 1 \text{ if vehicle } k \text{ has to reload at the crossdock}, k \in K \\ 0 \quad \text{otherwise} \end{cases}$$

s_i^k = the time at which the vehicle k leaves the node i $(i \in N, k \in K)$
t_k = the time at which vehicle k finishes unloading at the crossdock, $(k \in K)$
w_k = the time at which vehicle k starts reloading at the crossdock $k \in K)$
v_{ij}=*the time at which request* D_{ij} *is unloaded by its pickup vehicle at the cross dock* $(i \in P; j \in D)$.
In addition, M is an arbitrarily large constant.
The objective is to minimize the total cost of transportation.
Constraints consist on two types: The first one concerns vehicle routing while the second is related to the consolidation decisions at the cross-dock.
We noticed that both the pickup and the delivery parts can be formulated as VRPTWs, if the time windows were absent.
The main interest of our model was to provide a compact problem specification: the problem is NP-hard, so very small instances can be solved by commercial MIP solvers. Cplex solver was not able to find optimal solutions for instances with more than 12 nodes. This is why metaheuristics are studied in the sequel to tackle the problem in practice.

3- Problem resolution

This section presents the methods used for the resolution of the problem: A constructive heuristic and four metaheuristics: A MA, a TS, an ELS and an ILS. The obtained results are compared to choose the best one for our problem.

3.1 Constructive heuristic

A Best Insertion Heuristic (BIH) has been developed for the VRPCD. This heuristic starts with an empty tour. At each iteration, it calculates the minimum insertion cost for each customer previously untreated and performs better integration. When the remaining capacity is insufficient to accept new customers, a new tour is created. In our case, at each iteration of the BIH, the pickup node and its delivery nodes are respectively inserted in a collection and distribution tour. The cost of the insertion is evaluated for all possible positions in all trips and the best location is selected for each node. The feasibility of insertion is evaluated respecting the vehicle capacity and time windows. The specificity of the best insertion heuristic in this case is that the vehicle which serves the delivery node cannot start before all pickup vehicles returns to the cross dock. Otherwise, when inserting a pickup node, the feasibility of the insertion of the delivery node must be checked. In fact, for a given delivery node, if the last pickup vehicle returns too late to the cross dock, it may be impossible to serve it, even with a dedicated vehicle.
To check the time windows feasibility of the pickup part, we must go through all the nodes located after the location of the insertion and check if the offset of arrival dates of these nodes does not cause time windows violations. Solomon [4] introduced a method based on the storage of a number of variables to achieve these feasibility tests in O (1) instead of O (n). These techniques remain valid for the pickup part. However, for delivery, the insertion of a node can delay the arrival even for the nodes located before the insertion position, when the availability date of this delivery at the cross dock is greater than all availability dates of the other nodes of this trip. We introduced a technic inspired from the one of Solomon that allows us to do feasibility tests in O (1) even for this special case.
Figure 4 shows an example of 3 suppliers and 3 customers; it is assumed that S1 is the first one to be visited in the pickup part. The vehicle starts its trip from the cross-dock at 6 am and arrives after an hour at S1 (7 am). It starts it service at 7 and load 10 pallets, (its capacity is not yet reached). After 20 minutes the vehicle leaves S1 (Given that the fixed time for preparation is ten minutes and the time for loading each pallet is one minute, the total loading duration is 20 (10 + 10).
Before inserting a second node in the trip, we must check the following three conditions:
To generate a minimal cost.
The amount taken must respect the capacity of the vehicle.
Neither his time window nor those of its customers are violated.

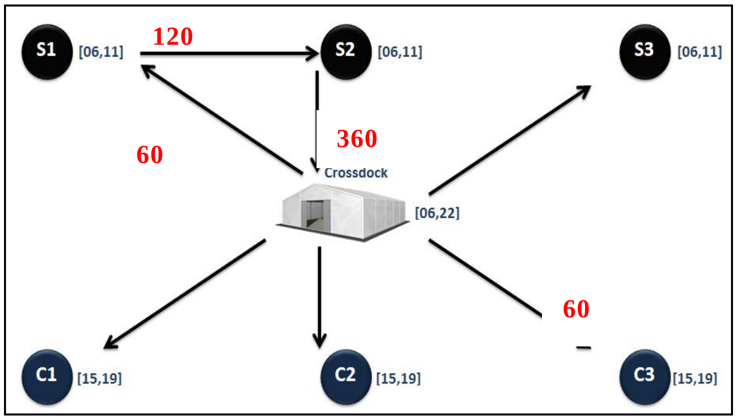

Figure 4.Example of insertion selection

To get to S2, the vehicle needs two hours; it arrives at 9:20 am and leaves at 9:40 am.
It remains to check if delivery trips are affected. Assume that the S2 delivers C3, the vehicle will need 6 hours to return to the cross dock, it gets there at 3:40 pm (9:40+6). As the vehicle will leave the cross dock at 4:10 pm (since the discharge and the reload at the cross dock will take 30 minutes) if the travel time between the cross dock and C3 is more than 3 hours and 50 minutes, the time window of C3 will not be respected. To summarize, before every insertion of a node, we must check the conditions mentioned before to ensure the feasibility of this insertion.

3.2The memetic algorithm for the VRPCD

Genetic Algorithm (or GA) is a metaheuristic proposed by Holland [5] in 1975 and popularized by Goldberg [6] in 1989. It is known that the basic version of the genetic algorithm is not aggressive enough on combinatorial optimization problems compared to other metaheuristics such as tabu search. This is why, Moscato [7] proposed a more powerful version; hybridized with a local search algorithm called memetics (Memetic Algorithm, MA). Local search in the memetic algorithm brings intensification. Indeed, the resulting solutions from the crossover are improved by local search before undergoing mutation.

3.2.1 Population, chromosomes and evaluation:

The proposed memetic algorithm starts with a population Pop containing ns chromosomes, sorted in increasing order of costs. The initial population includes random solutions and the BIH solution to introduce some intensification.
Each chromosome is coded as a list of suppliers followed by a list of customers. This list can be viewed as two giant tours which ignore vehicle capacity and time windows. The VRPCD solution is computed by an adaptation of a splitting procedure called split developed for the VRP by Prins[8] and described in next section.

{1 2 3 4 5 2' 1' 5' 4' 3'}

Figure 5 . Example of coding the VRPCD

3.2.2 Chromosome evaluation: Splitting procedure

In general, the principal of split for the VRP is to build an auxiliary graph H containing one dummy node 0 and n other nodes corresponding to n customers. Each subsequence of customers (Si, Si+1, . . . , Sj) corresponding to a feasible trip is modeled by a weighted arc (i−1, j) in H. The best VRP solution subject to the order given by the chromosome is obtained by computing a shortest path between node 0 and node n.
For the VRPTW, a chromosome is encoded as a sequence S containing the n customers, but not including tour delimiters. It may be seen as a giant tour where the vehicle capacity and time windows are ignored. Lack tour delimiters allows having chromosomes with the same length and using simple crossovers, like those used in the GA for the traveling salesman problem [14].
The best solution to the VRPTW, respecting the sequence , the capacity of vehicles and the time windows can be deduced by the splitting procedure: to calculate the shortest path in the auxiliary graph H containing a dummy node 0 and a node per customer. The shortest path from node 0 to any other node in the auxiliary graph H can be

calculated using Bellman algorithm for graphs without circuits. This procedure is used to determine where the giant tour should be cut to obtain feasible tours.
For the VRPCD, this operation is realized in two steps. In the first step, the auxiliary graph is built for the pickup part. This time each arc represents a trip which respects time windows and vehicle capacity, and the best pickup solution is deduced by computing the shortest path.
The principle of split in the pickup part is illustrated in Figure 5. The upper part of the figure shows an example of chromosome S = (S1, S2, S3, S4, S5). Numbers in parentheses indicate the amount to be collected and those in bracket represent time windows.

Node	a_i	b_i	r_i	Ridge	C_{ij}
CD	0	200	0	[CD,S1]	30
S1	0	50	15	[CD,S2]	25
S2	10	30	20	[CD,S3]	30
S3	20	60	20	[CD,S4]	40
S4	30	50	25	[CD,S5]	10
S5	80	120	20	[S1,S2]	20
				[S2,S3]	30
				[S3,S4]	20
				[S4,S5]	40

Table 2. Instance data used in Figure 6.

The auxiliary graph H is given in the middle of the figure, assuming vehicle capacity Q = 50. Each edge of the graph represents a feasible trip. Thus, the arc S1 models a dedicated trip for supplier S1, the value 60 is a go-and-back from the cross dock.
We note that the arc S1S2 is missing since the trip visiting S1 and S2 violates the time window of S2. The trip S2S3c and S4S5 are also feasible because neither the capacity of the vehicle none time windows are affected. All other trips that are not represented violate vehicle capacity or time windows of suppliers.
The result of the procedure split is shown in the lower part of Figure.

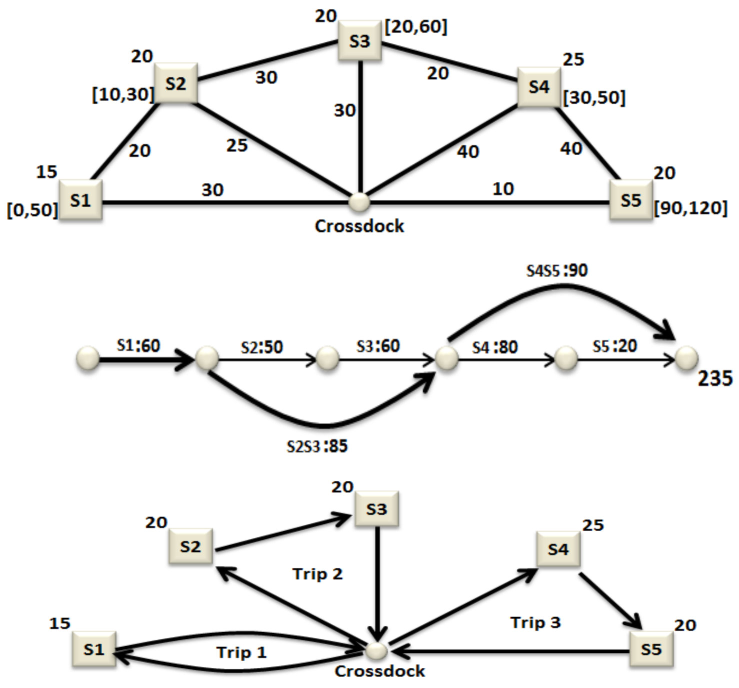

Figure.6Example of splitting procedure

In the second step, the auxiliary graph for the delivery part is built, another constraint is added this time, which is the availability of demands to be delivered, because delivery trips are constructed by taking into account availability dates of deliveries which depends on the arrival time to the cross dock of their pickup trips. Once the second auxiliary graph is built, the delivery solution is deduced in the same way as for the pickup one.

3.2.3 Selection and crossover

Each iteration of the memetic algorithm selects two parents from the population with the binary tournament: two chromosomes are randomly selected from the population and the best one is kept as the first parent P1. This process is repeated to get the second parent P2. These parents are then combined using the order crossover or OX, a classical crossover for the travelling salesman problem which randomly selects two cutting points i and j in the first parent. Customers between these two cutting points are copied to the same positions in the child. Missing customers are then added by scanning circularly parent P2 from the second relative position j + 1 to the position j.
The child is also filled circularly from j + 1. Crossover OX usually provides two children by inverting parents order. In our version, we generate a single child by randomly choosing the order of parents.
For the VRPCD, OX is applied separately to the pickup or to the delivery parts with a probability of 0.5. Numerical experiments proved that if both pickup and delivery parts are considered by the crossover, the resulting child is too different from its parents. Figure below shows an example of crossover OX applied to the delivery part.

P1:	C1	C2	|C3	C4|	C5
P2:	C2	C5	|C1	C4|	C3
C:	C5	C1	|C3	C4|	C2

Figure.7: Example of crossover OX for the delivery part

3.2.4 Local search:

Local search for the VRPCD is based on an adaptation of some classical VRP moves (Or-opt, exchange,...). For the pickup solution, if we consider the move or-opt that remove one supplier from a trip T1 and reinsert it in another trip T2, the arrival time to the cross dock of the vehicle of trip T2 may be delayed, and then it will shift the availability date at the cross dock of all the deliveries of this trip. This may make also the delivery impossible because of time windows. In such case, for each node of trip T2, if its new availability date at the cross dock will shift the departure time of its delivery trip, the feasibility of time windows must be checked for all the deliveries of this trip.
The improved child integrates the population and replaces one of the worst solutions to ensure the conservation of the best solution.

3.3Tabu search:

Tabu search (TS) is a meta-heuristic exploring the solutions space by moving at each iteration from the incumbent solution S to the best solution S' in its neighborhood N(S), even if the objective function deteriorates. To avoid coming back to recently visited solutions, a tabu list acting as short-term memory of visited solutions must be implemented. A diversification mechanism, based for instance on long-term memory, is often added to access unexplored regions of the search space. Numerous applications like job shop scheduling, graph coloring, TSP and other vehicle routing have been successfully tackled by TS, for mor details about applications can be seen in [15](Fellahi 2008). The proposed Tabu search starts from one initial feasible solution and improves it. This solution is built exactly like the initial solution in the MA: A best insertion heuristic is used to generate it. In the following the implemented TS is described by specifying its components, namely: Initial sampling, neighborhood, short-term memory, diversification mechanism and stop condition.
3.3.1 Initial sampling: The proposed TS starts from an initial feasible solution and improves it. This solution is built as the one used for the MA. A best insertion is used to find a first solution. Then, this solution is improved.
3.3.2 Neighborhood:Neighborhood structures are based on local movements: swap, 2-Opt, and insertion. The size of these neighborhoods is O (n2), involves a constant number of arcs exchanges. At each iteration of local search, all exchanges of arcs involving pickup or delivery trips are evaluated. Finally, realizable local movements are considered.
3.3.3 Short-term memory:The tabu list is the most used short-term memory to avoid looping back to already visited solutions. In its simplest form, a tabu list contains the solutions that have been visited in the recent past. Nevertheless, since recording a whole solution is space consuming, a better way of doing consists in memorizing a set of attributes able to characterize the solution.In the proposed neighborhood each successful move is saved in the tabu list for a number of iterations.
3.3.4 Diversification:The intensification phase consists of focusing the exploration in the promising regions, while the diversification phase consists of guiding the search into different regions less explored. In the literature there are

many ways for diversifying or intensifying the search of a TS method. The simpler and straightforward way of doing it consists of increasing or decreasing the size of the tabu list, as already explained. In our implementation the diversification was to remove the tabu list with a low probability to explore other research regions.
3.3.5Aspiration criterion:While central to TS, tabus are sometimes too powerful: they may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the searching process. It is thus necessary to use algorithmic devices that will allow one to revoke (cancel) tabus. These are called aspiration criteria. The simplest and most commonly used aspiration criterion, which is found in almost all TS implementations, consists in allowing a move, even if it is tabu, if it results in a solution with an objective value better than that of the current best-known solution (since the new solution has obviously not been previously visited). The key rule in this respect is that if cycling cannot occur, tabus can be disregarded. In our implementation, the same rule was applied, an aspiration criterion was to accept a move if it leads to a better solution than the best one already found.
3.3.6Stopping condition:In the TS we can stop the search at any time; there are several possible stopping conditions:
-If a proven optimal solution was found.
-If a limit has been reached concerning
-The Number of iterations;
-The Calculation time.
-If the research seems to stagnate: iterations without improvement of the best found solution best configuration found.
In our case we choose as a stopping condition a fixed number of iterations.Algorithm X presents the TS implemented in this paper.

Algorithm 1 – Pseudo-code of the TS used
```
• Generate an initial solution by the best heuristic insertion S0 ; S := S0
• S* := S ; f* := f(S)
• T := {} // Tabu list
• r float between 0 and 1
• Repeat
            Repeat
                m := the best move among non tabu and exceptional tabu moves (aspiration criterion)
                S := S (+) m
            While m exists
        _ Give a random value to r
        – if f(S) < f(S*) do S* := S ; f* := f(S);add m to T ;
        _ else if r<0.01 do T := {}
        _ end if
        _ randomly generate S
 • Until a stopping criterion is reached
  • Return S*
```

3.4An ILS and ELS for the vehicle routing problem with cross docking.

ILS is a simple metaheuristic that needs only an initial solution, local search and perturbation mechanism. Its overall structure is described by the algorithme 2. First, a constructive heuristic is used to generate a starting solution, which is converted to a local optimum by the local search. Then the main loop of the algorithm executes ngen iterations or generations. At each iteration, a copy of the current best solution is changed randomly with the disruption procedure (random move), acting as the mutation operator of genetic algorithms. The resulting solution is then improved by local search and replaces the best solution only in case of improvement.

Algorithm 2 – Pseudo-code of the ILS used
```
1: Initialize the random generator
2: Heuristic (BestSol)
3: LocalSearch (BestSol)
4: for gen := 1 to ngen do
5: Sol := BestSol
6: Perturb (Sol)
7: LocalSearch (Sol)
8: UpdateSolution (Sol,BestSol)
9: end for
10: Return (BestSol)
```

The pseudo-code of the metaheuristic ELS is given by the algorithm 3. Starting with a solution built by a heuristic and improved by a local search, the ELS perform ngen generations. Each generation build Nchild children-solutions. Each child is obtained from a copy of the current BestSol solution, which is then modified by the perturbation procedure and improved by local search. The best solution is saved in child-BestChildSol.

If at the end of a BestChildSol generation is better than the current solution BestSol, it is then replaced by BestChildSol for the next generations.

Algorithm 15 – Pseudo-code of the ELS used

```
 Initialize  the random generator
2: Heuristic (BestSol)
3: LocalSearch (BestSol)
4: for gen := 1 to ngen do
5: for child := 1 to nchild do
6: ChildSol := BestSol
7: Perturb (ChildSol)
8: LocalSearch (ChildSol)
9: UpdateSolution (ChildSol,BestChildSol)
10: end for
11: UpdateSolution (BestChildSol,BestSol)
12: end for
13: Return (BestSol)
```

4- Computational results

Our algorithms were implemented in Java and executed on a PC with a dual processor *AMD Turion* 64, 2 *GHZ*, and 3 *GB* of *RAM*.

4.1 Data

Instances were generated randomly while ensuring non-violation of time windows; this is why we have chosen to divide the day into two parts: Morning (6 am to 11 am) is reserved for pickup and evening (3 pm to 7 pm) is for delivery.

Taking into account the arrival time of the last pickup vehicle at the cross dock, the unload/load time and the travel time between the cross dock and the delivery node, the delivery vehicle must arrive before the end of the customer's time window.

The test data consist of three Euclidean sets, each one containing 5 instances denoted by Ia, Ib, Ic, Id and Ie, respectively, where I stands for the number of suppliers. Each instance has the same number of suppliers and customers known by their pickup and delivery locations (*x; y*). The time window for each pickup node is limited between 6:00 and 11:00 am. For delivery it is between 3:00 and 7:00 pm. The time horizon for the whole transportation operation is from 6:00 to 22:00. The demand transported from each pickup location to delivery location is given in number of pallets. Vehicles drive at a constant speed of 60 km/h and have a capacity of 33 pallets. It takes ten minutes to prepare a vehicle, plus an additional one minute for each pallet to be loaded or unloaded.

FILE	CLIENT	BIH	MA	TIME	TS	Time
DATA4_A	8	1070,446	1069,811	0,49	1069,81	0,6
DATA5_A	10	1201,924	1143,929	0,32	1143,929	0,5
DATA5_B	10	1047,756	994,631	0,37	994,631	1
DATA6_A	12	1358,065	1286,179	0,31	1286,179	1
DATA6_B	12	1447,332	1403,315	0,35	1403,315	1
DATA10_A	20	2394,807	1906,224	0,85	1897,972	1
DATA10_B	20	2799,268	**2276,194**	0,53	2376,118	1
DATA10_C	20	2612,165	**1856,265**	0,78	2223,829	1
DATA10_D	20	2540,801	**2239,171**	1,34	2361,934	0,4
DATA10_E	20	2613,486	**2029,263**	0,74	2290,783	1

DATA10_F	20	2204,1	**1790,135**	1,06	1869,031	1
DATA10_G	20	2232,782	**2000,594**	0,81	2002,918	1
DATA10_H	20	2313,331	1956,153	1,26	**1905,842**	1
DATA10_I	20	2466,807	2075,5	0,96	**2037,867**	1
DATA10_J	20	2630,447	**1948,906**	0,77	2074,641	1
DATA30_A	60	9966,764	**7474,786**	4,09	7718,219	1
DATA30_B	60	9468,676	**7555,811**	2,11	7676,292	1
DATA30_C	60	8706,521	**6148,789**	6,49	6657,493	1
DATA30_D	60	9072,882	**7042,106**	6,71	7158,756	1
DATA30_E	60	8351,863	**5986,499**	4,96	7100,559	2
DATA30_F	60	9806,669	**7131,365**	5,33	7782,413	2
DATA30_G	60	9034,319	**6774,495**	2,54	6889,669	2
DATA30_H	60	9195,169	**7062,248**	3,43	7309,926	2
DATA30_I	60	8778,929	**6863,495**	4,07	7297,374	2
DATA30_J	60	8150,426	**6303,393**	5,83	6386,244	2
DATA50_A	100	14691,297	**11096,202**	7,06	11529,803	2
DATA50_B	100	14773,067	**10796,803**	11,62	11110,912	2
DATA50_C	100	14782,854	**11281,22**	11,06	11577,424	4
DATA50_D	100	14787,624	**10831,557**	6,95	11537,786	4
DATA50_E	100	14261,985	**10445,996**	10,68	10361,210	5
DATA50_F	100	14831,501	**10998,258**	10,7	11721,652	5
DATA50_G	100	15107,637	**10675,237**	17,71	11698,976	5
DATA50_H	100	14690,954	**10574,697**	9,23	10956,072	6
DATA50_I	100	14778,554	**10129,632**	11,04	11253839	6
DATA50_J	100	14283,004	**10008,955**	10,86	10642,438	6

Table2.Comparaison between TS,memetic algorithm and BIH**.**

The first tests confirm the efficiency of the memetic algorithm which improves strongly the results of the Best Insertion Heuristic, and is better than the TS for the most instances with reasonable computational times (less than 16 seconds for all the instances). The ELS and ELS tests are in process to compare with the current methods. Cplex was able to find results for small instances (up to 12 nodes), after it gets out of memory, that is why we did not mention the results in the table above. We precise that all metaheuristics gave optimal solutions for these small instances.

5- Conclusion

Although cross-docking has been widely practiced within both manufacturing and retailing companies and brings benefits to companies, there are very few studies on the integration of vehicle routing problems and cross-docking. In this study, a heuristic and four metaheuristics are used to solve the vehicle routing problem with cross docking. Very good results are obtained and other tests are outstanding on large instances with the introduction of several improvements. Additional constraints will also be considered in future work, such as direct connections (suppliers to clients without going through the cross dock) when it is gainful. A lower bound will be also developed to better evaluate the performances of the proposed methods.

References

[1] M.Wen, J.Larsen, J.Clausen, J.F Cordeau, and G Laporte, Vehicle routing with Cross-Docking, Journal of Operational Research Society ,vol 60 (2008),pp 1708–1718.

[2] Y. H Lee, W. J Jung, and K.M Lee, Vehicle routing scheduling for cross docking in the supply chain, Computer and Industrial Engineering, vol 51 (2006), pp.247–256.

[3] F.A Santos,G.R Mateus, and A.S Da Cunha ,A novel column generation algorithm for the vehicle routing problem with cross-docking. INOC'11 Proceedings of the 5th international conference on Network optimization, (2011), pp 412-425.

[4] Solomon, M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper.Res.35 254–265.

[5] J.H. Holland. Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor, MI, USA, 1975.

[6] D.E. Goldberg. Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Reading, MA, USA, 1989.

[7] P. Moscato. Memetic algorithms: a short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, New ideas in optimization, pp 219–234. McGraw-Hill, 1999.

[8] C. Prins, A simple and effective evolutionary algorithm for the vehicle routing problem, Computer Operations Research, vol 31(2004),pp 1985–2002.

[9] Sung, C.S., Song, S.H., 2003. Integrated service network design for a cross-docking supply chain network, Journal of the Operational Research Society, 54, 1283-1295.

[10] Lee, Y.H., Jung, J.W., Lee, K.M., 2006. Vehicle routing scheduling for cross-docking in the supply chain, Computers & Industrial Engineering, 51, 247-256.

[11] Apte,M.U., Viswanathan, S.: Effective cross docking for improving distribution efficien-cies.International Journal of Logistics: Research and Applications 3, 291–302, (2000)

[12] N.Labadi ,C.Prins,M.Reghioui. A memetic algorithm for the vehicle routing problem with Time windows, RAIRO Operations Research

[13] Musa R, Arnaout J P, Jung H. Ant colony optimization algorithm to solve for the transportation problem of cross-docking network. Computers &Industrial Engineering2010; 59 (1):85–92.

[14] M.Reghioui. Problèmes de tournées de véhicules avec fenêtres horaires ou préemption des taches. Thèse de doctorat de l'Université de Technologie de Troyes.185 p.