A HYBRID MULTISTART HEURISTIC FOR THE PICKUP AND DELIVERY PROBLEM WITH AND WITHOUT TRANSSHIPMENT

Rodrigue Tchapnga Takoudjou, Jean-Christophe Deschamps, Rémy Dupas

9^{th} *International Conference of Modeling, Optimization and Simulation - MOSIM'12*
June 6-8, 2012 - Bordeaux - France
"Performance, interoperability and safety for sustainable development"

A HYBRID MULTI-START HEURISTIC FOR THE PICKUP AND DELIVERY PROBLEM WITH AND WITHOUT TRANSSHIPMENT

Rodrigue Tchapnga Takoudjou, Jean-Christophe Deschamps, Rémy Dupas

Univ. Bordeaux, IMS, UMR 5218, F-33400 Talence, France.
CNRS, IMS, UMR 5218, F-33400 Talence, France.
33405 Talence cedex - France
Tel: 335-400-03625; e-mail: (rodrigue.tchapnga-takoudjou, Jean-Christophe.Deschamps, remy.dupas)@ims-bordeaux.fr

ABSTRACT: *The pickup and delivery problem (PDP) is a special class of transportation problems. The objective of the PDP is to satisfy at minimum cost a set of transport requests while respecting a set of constraints. A variant of the PDP is the pickup and delivery problem with transshipment (PDPT). In PDPT, a request can be satisfied by more than a vehicle. For instance, one vehicle picks up a load at the supplier location, drops it off at a transshipment point with short storage and another vehicle carries and delivers this load to the final destination. In this paper, we propose a multi-start heuristic with path relinking (PR) and variable neighbourhood descend (VND) to solve the PDP and we use a local search based on transshipment to optimize the solution of the PDP. Our approach is tested using benchmark instances from the literature.*

KEYWORDS: *Pickup and delivery problem, transshipment, Multi-start heuristic, Path relinking, Variable neighbourhood descend.*

1 INTRODUCTION

In a sluggish economy and increased competition, companies try to optimize their supply chain in general and transportation activity in particular. Transport plays a major and important role in the performance of the supply chain. For this reason, companies need to develop innovative, efficient and effective approaches for the management of their transportation activities. To meet this challenge, several research lines on the issue of transportation studied as an important contribution to the supply chain performance have emerged. On technological plan, we can quote the development of initiatives regarding ICT (Information and Communication Technologies) such as RFID (Radio Frequency Identification), GPS (Global Positioning System) promote effective traceability of products during transportation. On operational plan, research initiatives are more oriented on routing problems including cross-docking (Boysen and Fliedner, 2010) and transshipment as an efficient way to provide flexibility in the organisation of transportation activities.

Transshipment is a practice used to optimize the routing of products to their address by allowing a demand (request) of transport to be satisfied by more than one vehicle. For instance, one vehicle picks up a load at the supplier location, drops it at a transshipment point with short storage (warehouses) and another vehicle carries and delivers this load to the final destination. This practice is particularly used in transportation systems with "cross-docking", where the objective is to reduce the storage time of products at the distribution centre by better coordinating the pickup of products upstream, and the distributions of those products downstream.

Transshipment remains, however an expensive operation; a consumer of time and resources. Nevertheless, it but offers certain flexibility while authorizing a reduction of the total distance travelled, the number of used vehicles, and by projection in the field of the sustainable development, a better control of the carbon footprint associated with a transported load.

By starting from this premise, the objective of this paper is to propose a multi-start heuristic with path relinking (PR) to solve the PDP with and without transshipment in order to minimize primarily the number of vehicles used and secondarily the total travelled distance. At each iteration of the multi-start heuristic a PDP solution is computed. Then, this solution is destroyed and repaired to obtain PDPT solution. The destruction process and the repair process is a local search based on transshipment. The best

solution found during all iterations of the multi-start heuristic is returned as the solution of the problem.

This paper is structured as follows: In section 2, we present a literature review and we discuss some previous works related to the PDP and the PDPT. Section 3 is dedicated to the formalization of the PDP and the PDPT. Section 4 presents the multi-start heuristic with variable neighbourhood descent and path relinking approaches to solve the PDP and PDPT. Section 5 shows experimental results, their analysis and points out some concluding remarks.

2 LITERATURE REVIEW

The pickup and delivery problem (PDP) addresses the construction of optimal routes to satisfy transportation requests (each request is defined by a couple of pickup and delivery points), precedence constraints (it is mandatory to pick up before delivering), vehicle capacity constraint, pairing constraint (each request is executed by the same vehicle), and time windows (PDPTW) constraints.

A mathematical model and an exact algorithm based on a column generation scheme with a constrained shortest path as the sub problem was proposed by (Dumas et al., 1991) to solve the PDPTW. A more recent and interesting survey for the PDPTW was offered by (Parragh et al., 2008a, Parragh et al., 2008b). Pickup and delivery problem is also known to be a NP-hard combinatorial optimization problem (M.W.P. et al., 1995). For this reason, exact methods are restricted to optimally solve only small size problems. To overcome this difficulty, for many years, researchers develop heuristics and metaheuristics capable of giving solutions of good quality in reasonable time. (Lu and Dessouky, 2006) have developed a parallel construction heuristic for the PDPTW. Their main contribution was to define new criteria to evaluate requests insertion based on reduction of time slack comparing to the classical one based on the incremental distance measure. (Li and Lim, 2001) have developed a hybrid metaheuristic based on tabu search and simulated annealing to solve the PDPTW. They have also generated several test instances for the PDPTW. A two stage hybrid algorithm for the PDPTW was designed by (Bent and Hentenryck, 2006). To reduce the number of used vehicles, simulated annealing (SA) is used at the first stage and large neighbourhood (LNS) algorithm is applied at the second stage to decrease total travelled cost.

The PDPT is a variant of PDP such that each request can be served by more than one vehicle (pairing constraint is relaxed) by dropping a good at a transfer point and picking it up by another vehicle. There is little literature on the PDPT. (Cortés et al., 2010) have proposed a MIP model for the dial-a-ride version of the PDPT. The authors have also implemented a branch-and-cut able to handle any number of transfer points and solved to optimality instances with up to six customers and one transshipment point. (Mues and Pickl, 2005) have developed a column generation approach for a particular case of the problem and they solved instances with up to 70 requests. (Mitrovic-Minic and Laporte, 2006) have developed a two phase heuristic algorithm for the PDPT with time windows; the first phase is a greedy construction using cheapest insertions, and the second one is an improvement phase based on request reinsertions. The heuristic was launched with instances involving up to 100 requests, and results have showed that significant savings can be obtained if the requests are clustered. (Shang and Cuff, 1996) have worked on multi objective PDPT where patient record, equipment, and supplies can be transferred between vehicles. The objectives are to minimize vehicle expense, tardiness and travel time. They developed a look ahead heuristic to solve the PDPT. Firstly, mini-routes based on shipment request are constructed and then incorporate into vehicle routes with a special insertion procedure. (Qu and Bard, 2012) were the first to propose a metaheuristic based on the GRASP and adaptive large neighbourhood search (ALNS) for the pickup and delivery problem with transshipment. In the construction phase of the GRASP, they used a sequential insertion procedure to insert new request on route and the improvement phase was based on the ALNS developed and proposed in (Ropke and Pisinger, 2006).

Multi-start methods are typically used to solve combinatorial problems. Multi-start methods have two phases (Martí et al., 2010): the first one in which the solution is generated and the second one in which the solution is improved. At the end of each global iteration a local optima solution is produces and the best overall is keep as the final solution. This approach was successfully applied to resolve several combinatorial problems (Crainic et al., 2011, Bräysy et al., 2004, Boese et al., 1994, Brønmo et al., 2007).

To clarify and define our contribution following the analysis of the state of the art, we make the following assumptions:

- Vehicle fleet is homogenous, and vehicle capacity is bounded.
- The number of vehicle is not fixed.
- Travel time between each location is symmetric.
- Each location is associated with a time window.
- If a vehicle arrives before the earliest pickup or delivery time of a customer, it is allowed to wait until the start of the time window.

- Each pickup location has to be visited before visiting its corresponding delivery location.
- The pickup location and the delivery location are visited by the same vehicle, unless the corresponding request is transhipped between vehicles.
- The transshipment incurs additional travel unloading and loading time/cost.
- Each location can be used as a transshipment point.
- A vehicle cannot load and unload the same product in the same transshipment point.

3 PROBLEM FORMULATION

This section presents different notations and concepts used to describe the PDP and the PDPT.

3.1 Formulation of the PDP

Let n be the number of customers and let $N = P \cup D$ a set of $2n$ elements partitioned into two subsets: $P = \{1, ..., n\}$ the set of pickup nodes and $D = \{1+n, ...2n\}$ the set of delivery nodes. Each customer request is characterized by a pair of nodes $(i,\, i+n)$. The origin or pickup node of the request is represented by $(i \in P)$ and the destination or delivery node is represented by $(i+n) \in D$. We define $R = \{(i,\, i+n),\, i \in P,\, i+n \in D\}$ as the set of requests. Any node $j \in N$ is associated with a time windows $[a_j,\, b_j]$. To satisfy the request $(i,\, i+n)$ of a customer $i \in n$, a positive quantity l_i of products is collected at node $i \in P$, and a quantity l_{i+n} of products is delivered at $(i+n) \in D$ such that $l_i = -l_{i+n}$. We define s_j as the service lead time at each pickup or delivery node $j \in N$. The set of homogeneous vehicles is denoted K and each vehicle has a capacity Q. The depot $o = o_1 \cup o_2$ from where vehicles start and end their routes is modelled by two nodes o_1 and o_2 representing the origin and destination of each vehicle route.

The PDP is characterized and defined by a graph $G = V \cup E$ such as: $V = N \cup o_1 \cup o_2$ is the set of vertices, $E = V \times V$ is the set of edges. Between each couple of vertices $(j_1,\, j_2) \in V$ there exists both a nonnegative distance $d_{j_1 j_2}$ and a travel time $t_{j_1 j_2}$. $x^k_{j_1 j_2}$ is a binary variable used to specify if edge $(j_1,\, j_2) \in E$ is taken by vehicle $k \in K$. It is equals to 1 if true and 0 otherwise; dt^k_j is a real variable used to specify the time at which vehicle $k \in K$ reaches the customer $j \in V$; L^k_j is used to specify the load of vehicle $k \in K$ upon servicing customer $j \in V$.

The main constraints that a solution of PDP has to satisfy are time windows constraints:

$$dt^k_j \in [a_j,\, b_j]\,,\ \forall j \in V,\, k \in K \tag{1}$$

vehicle capacity:

$$L^k_j \leq Q,\ \forall j \in V,\, k \in K \tag{2}$$

precedence constraints: it is mandatory to pickup before delivery:

$$dt^k_i \leq dt^k_{i+n},\ \forall i \in P,\, k \in K \tag{3}$$

and pairing constraints:

$$\sum_{j \in N} x^k_{ij} - \sum_{j \in N} x^k_{(i+n)j} = 0,\ \forall i \in P,\, k \in K \tag{4}$$

3.2 Formulation of the PDPT

To formalize the PDPT, we keep the notations previously introduced, and we add the following variables and parameters: Let $T\,(|T| \geq 1)$ be the set of transshipment points. Each transshipment point $t \in T$ is split into two separate nodes, e_t (start node) and s_t (finish node). If the request of customer $i \in n$ has to be transhipped, when a vehicle passes through transshipment point $t \in T$, it first enters node e_t to drop a quantity l_i, $i \in N$ that must be transferred to a different vehicle. The vehicle then proceeds to node s_t, where others products possibly on hold are loaded into the vehicle. We call e_t the inbound door for the transshipment point t and we denote by s_t it outbound door. We define $B \geq 0$ as the unit time for unloading and reloading product (or a pallet) at the transshipment point. The PDPT is then characterized and defined by a graph $G = V \cup E$ such as: $V = N \cup o \cup T$ is the set of vertices, $E = V \times V$ is the set of edges. Every solution must satisfy each transport request and constraints such that each route starts at o_1 and finishes at o_2. The main constraints that a solution of PDPT has to satisfy are identical to those used in PDP, except precedence constraint at the transshipment points. For instance, if request $(i, i+n)$ is executed by two vehicles through the use of transshipment point t, then vehicle $k_1 \in K$ first pickups l_i at i, delivers it to e_t before vehicle $k_2 \in K$ reloads l_i at s_t and transports it to customer $n+i$.

In the following, we use the following notations. H: total number of vehicles used to satisfy the requests; $r\,(k)$: route of vehicle (sequence of stop locations assigned to one vehicle) $k \in K$; $ND\,(k)$: total number of nodes visited by a vehicle $k \in K$; $Rcost\,(k)$: total travel distance of route $k \in K$; $cost\,(i,\, i+n)$: insertion cost of request$(i,\, i+n) \in R$; Sol: PDP solution; $TSol$: PDPT solution TD: total travelled distance.

Within the framework of this work, we used the evaluation function defined by (Bent and Hentenryck, 2006). Let $F(Sol)$ be the evaluation function.

$$F(Sol) = \alpha H + \beta TD - \gamma \sum_{k=1}^{H} ND(k)^2 \quad (5)$$

The constants α, β, γ are weights assigned to each term in the evaluation function, and $(\alpha, \beta, \gamma) \in [0, 1]$, $\alpha+\beta+\gamma = 1$. As the objective consists in minimizing primarily the number of vehicles and secondarily the total travelled distance, we choose $\alpha > \beta > \gamma$. The first component consists of minimizing the number of vehicles used. The second part consists of minimizing the total travelled distance. The third one is the square of number nodes visited by each vehicle during it tour. The purpose of this last component is that each vehicle must serve as much as possible the maximum of transport requests in the same route. This allows during the use of different neighbourhoods to promote the solutions using a minimum number of vehicles.

4 HYBRID HEURISTIC FOR THE PDP

PDP and PDPT are both hard combinatorial optimization problems. The heuristics for such problems must avoid being trapped by local optimum. To overcome local optimality, a diversification procedure is needed. We can obtain diversification by re-starting local search based on Variable Neighbourhood Descent (VND) and an intensification procedure using Path Relinking (PR), from a new solution once a solution space region has been explored. Indeed, for every calculation step, requests are randomly ordered. According to this order, every request is introduced into the solution under construction by a constructive and parallel greedy heuristic. The proposed heuristic is considered as hybrid because it combines several techniques and methods issued from different metaheuristics such as: path relinking, variable neighbourhood descent. The solving principle is based on the following steps:

- Random ordering of requests.
- An initial PDP solution (Sol) is calculated by gradually inserting the requests in routes associated to vehicles.
- The Variable Neighbourhood Descent method is applied to improve the initial solution, before using Path Relinking.
- Transshipment is then used to "destroy" and "repair" PDP solution to obtain a better solution (PDPT solution).
- The whole procedure is restarted until a maximum number of iterations is reached. Overall PDP solutions and PDPT solutions are compared and the best one corresponds to the heuristic's output.

The general principle of the hybrid multi-start heuristic is detailed in the pseudo-code described by the algorithm 1.

Algorithm 1: MULTISTART

input : Set of request R; Paramater $MaxIter$; Paramater λ; Set of Transfer point T
output: Sol

$Pool \longleftarrow \emptyset$; $Sol \longleftarrow \emptyset$; $Sol_1 \longleftarrow \emptyset$; $F(Sol) \longleftarrow \infty$;
for $i \longleftarrow 0$ **to** $MaxIter$ **do**
  $Random(R)$;
  foreach *request* $(i, i+n)$ *in* R **do**
    $Sol_1 \longleftarrow PARA((i, i+n), Sol_1)$;
  end
  $Sol_1 \longleftarrow VND(Sol_1)$;
  if $i < \lambda$ **then**
    $Pool \longleftarrow Pool + \{Sol_1\}$;
  end
  $Sol_g \longleftarrow Guiding(Pool)$;
  $Sol_i \longleftarrow Sol_1$;
  $Sol_i \longleftarrow PATHRELINKING(Sol_i, Sol_g)$;
  if $F(Sol_i) < F(Sol_1)$ **then**
    $Sol_1 \longleftarrow Sol_i$;
  end
  $UPDATE(Pool, Sol_1)$;
  $Sol_1 \longleftarrow TRANSSHIPMENT(R, Sol_1, T)$
  if $F(Sol_1) < F(Sol)$ **then**
    $Sol \longleftarrow Sol_1$;
  end
  $Sol_1 \longleftarrow \emptyset$
end
Return Sol ;

Each complex operation, constituting the multi-start heuristic and represented here by a function will be considered separately and described in detail in the following.

4.1 Random order of Requests

As we have stated above, since every request possesses a time windows, a service time and because the capacity of vehicles is limited, the order in which every request is executed has an impact on the quality of the solution. This process allows us to have diversity. In order to avoid having very similar solutions, and guarantees a certain degree of difference we use

memory mechanism.

4.2 Initial solution for the PDP

The proposed constructive heuristic operates in a parallel and greedy way to insert a request. The initial solution $Sol = \{r(a)\}$ is consisting of a single route containing only the starting point and the ending point of each route ($r(a) = \{o_1, o_2\}$). Requests are then successively introduced into the tour of a vehicle offering the minimal increase of the cost of transport. If no vehicle can satisfy a request because of the non compliance with constraints (vehicle capacity, time windows, etc.), a new route is created to welcome the considered request. The pseudo-code of the parallel constructive algorithm for the PDP is shown in the algorithm 2.

Algorithm 2: PARA

input : Request $(i, i+n)$; Sol
output: Sol; $Cost(i, i+n)$

```
1  Cost(i, i+n) ⟵ ∞;
2  route ⟵ ∅ ;
3  foreach r(k) in sol do
4  |  A ⟵ The cost of the best insertion of i
   |  and i+n in r(k) ;
5  |  if A < Cost(i, i+n) then
6  |  |  Cost(i, i+n) ⟵ A ;
7  |  |  route ⟵ r(k) ;
8  |  end
9  end
10 if Cost(i, i+n) ≠ ∞ then
11 |  Insert request (i, i+n) at the best
   |  insertion slots of route ;
12 else
13 |  Create new trip r(h) ;
14 |  Cost(i, i+n) ⟵ The insertion cost of
   |  (i, i+n) in r(h) ;
15 |  Insert (i, i+n) at the best slots in route
   |  r(h) ;
16 |  Sol ⟵ Sol + {r(h)} ;
17 end
18 Return Sol; Cost(i, i+n) ;
```

4.3 Variable neighbourhood descent (VND)

The VND is an improved local search described in (Hansen and Mladenović, 2001). The VND is used to explore the neighbourhood of the current solution, which is based on three operators defined below: N_1(ADR), N_2(RNR), and N_3(SWR). With these three operators, we can explore the solution space more intensively. An operator is a move that transforms one solution to another with small modifications. The VND is described in algorithm 3.

In Algorithm 3 (ligne 4), $LS(X, N_i)$ refers to the local search procedure in the operators defined by N_i. The VND used best improvement local search for RNR and ADR. For ADR the VND used first improvement local search. In best improvement operators the whole neighbourhood is analysed and the best solution is kept. In first improvement local search the first better solution found in the neighbourhood is keep. In the following, we described the three neighbourhoods used to construct the VND.

Algorithm 3: VND

input : Set of Neighborhood structure N_n $(n = 1, ..., 3)$; Initial Solution X
output: X

```
1  repeat
2  |  n ⟵ 1 ;
3  |  repeat
4  |  |  X' ⟵ LS(X, N_i);
5  |  |  if F(X') < F(X) then
6  |  |  |  X ⟵ X';
7  |  |  else
8  |  |  |  n ⟵ n + 1;
9  |  |  end
10 |  until n = 3;
11 until No improvement is obtained;
12 Return X;
```

SWR (swap requests between routes): In SWR neighbourhood, two requests belonging to two different routes are exchanged together provided that all PDP constraints are satisfied. If $(i, i+n) \in r(a)$ and $(j, j+n) \in r(b)$ with $a \neq b$, withdraw $(i, i+n)$ from $r(a)$ and $(j, j+n)$ from $r(b)$. Insert $(i, i+n)$ at the best slots of $r(b)$ and $(j, j+n)$ at the best slots $r(a)$. Figure 1 depicts SWR moves. In figure 1, request 2 and 5 are swapped with the aim to reduce travelled costs.

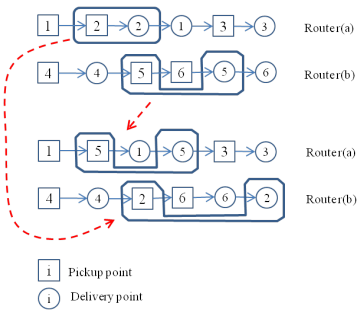

Figure 1: SWR moves

RNR (remove and insert a request): In RNR neighbourhood, provided that all PDP constraints are satisfied, request belonging to one route is removed and inserted in another route. Suppose

that we have two routes $r(a)$ and $r(b)$ such that $(i, i+n) \in r(a)$, with RNR we remove $(i, i+n)$ from $r(a)$ and insert it at the best slots to $r(b)$. In figure 2, request 2 is removed from its current route and inserted into another route with the aim to reduced total number of vehicles used.

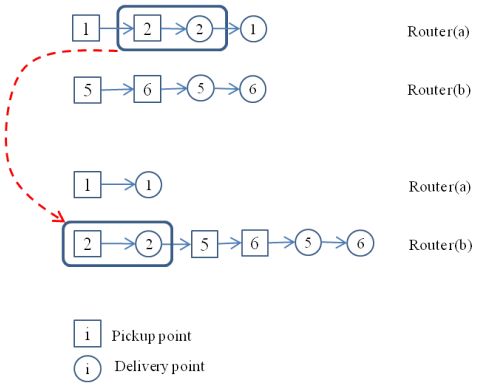

Figure 2: RNR moves

ADR (advance or delay a request): In ADR neighbourhood, requests belonging to one given route is advanced or delayed in the same route if PDP constraints are all satisfied. In figure 3, the execution of request 1 is delayed while the execution of request 3 is advanced with the aim of reducing total travelled distance.

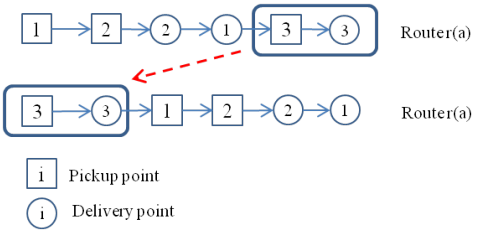

Figure 3: ADR moves

4.4 Path relinking

Path relinking (Martí et al., 2006, Resende et al., 2010) is a method where solutions of a combinatorial problem are generated by combining elements of other solutions. As noted by (Ho and Gendreau, 2006), path relinking operates on the basis of three components: reference set, initial and guiding solutions, neighbourhood structure for moving along paths.

Reference set of elite solutions or pool of elite solutions is initializing during the first λ iterations of the heuristic. While the number of iterations is less than λ, the best PDP solution provided by the VND is automatically integrated in the pool. For the remaining iterations, if the solution provided by the VND is better than the worst solution in the pool, the pool is updated by replacing the worst solution by the new one provided by the VND. We used λ as a parameter to define the length of the pool and to ensure the diversity of solutions. The value of is adjusted during the computational phase.

To build a path, the PDP solution obtained in each iteration is used as the initial solution and a guiding solution is randomly chosen in the pool. Once the initial solution and the guiding solution are identified, we progressively transform the initial solution (I) into a guiding solution (G). For example, in figure 4, the objective is to minimize a cost function. The path from I (initial solution) to G (guiding solution) using red line provide one solution improving I but not G while the path with green line provide three better solutions.

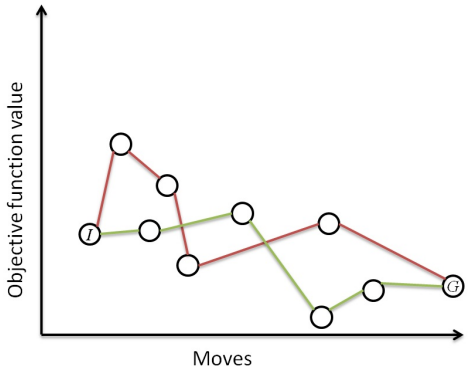

Figure 4: the technique of path relinking

The transformation process is based on a constructor operator called RouteMap. RouteMap is a procedure that takes as parameters two routes $r(a)$ and $r(b)$, then gradually transformed $r(a)$ into $r(b)$. This operator uses the following movements: remove (remove a node from a route), replace (replace one node in a route with another node), and insert (insert a node in a route). Table 1 shows how RouteMap works. Suppose that two routes $r(a) = \langle x, b, c, d, e, f, g \rangle$ and $r(b) = \langle b, c, f, x, m, o \rangle$.

$r(b)$	Move apply on $r(a)$	$r(a)$ in progress
b	remove x	$bcdefg$
c	no movement	$bcdefg$
f	remove c	$dbcfg$
x	replace g by x	$bcfx$
m	insert m	$bcfxm$
o	insert o	$bcfxmo$

Table 1: RouteMap

At each step of the transformation process, when a node is removed from $r(a)$, the node with which it is coupled by a request is also removed. The corresponding request is then reinsert in the solution on which $r(a)$ belongs. The obtained solution undergoes a local search. The pseudo-code of the transformation process is depicted in algorithm 4.

Algorithm 4: PATHRELINKING

input : I: Initial solution; G : Guiding solution
output: Sol_1;

foreach *route* $r(k_1)$ *in* G **do**
  foreach *route* $r(k_2)$ *in* I **do**
    $Sol_1 \longleftarrow RouteMap(r(k_1), r(k_2))$
  end
end
Return Sol_1 ;

4.5 The transshipment heuristic

At each iteration, the transshipment heuristic is used to improve the PDP solution and obtain a solution to the PDPT. From the current PDP solution, each request $(i, i+n) \in R$; is removed from the solution. Then, $(i, i+n)$ is split into two different requests (i, e_t) and $(s_t, i+n)$, where e_t and s_t are the inbound/outbound doors of a transshipment point $t \in T$. The best reinsertion cost of $(i, i+n)$ in the solution is computed. The best insertion cost to insert (i, e_t) following by insertion of $(s_t, i+n)$ in the solution is computed. The cost to insert $(s_t, i+n)$ following by the insertion of (i, e_t) at their best position is computed. Between the three possibilities, the insertion or the reinsertions offering the minimum cost is performed. The local search based on transshipment principle is presented in algorithm 5.

Algorithm 5: TRANSSHIPMENT

input : R; Sol; T;
output: $TSol$;

$TSol \longleftarrow Sol$;
$Tbest \longleftarrow F(Sol)$;
foreach $(i, i+n)$ *in* R **do**
  $Sol_1 \longleftarrow Sol - \{(i, i+n)\}$;
  foreach t *in* T **do**
    $Sol_2 \longleftarrow Sol_1$, $Sol_3 \longleftarrow Sol_1$, $Sol_4 \longleftarrow Sol_1$;
    $Sol_2 \longleftarrow PARA((i, i+n), Sol_2)$;
    $Sol_3 \longleftarrow PARA((i, e_t), Sol_3) + PARA((s_t, i+n), Sol_3)$;
    $Sol_4 \longleftarrow PARA((s_t, i+n), Sol_4) + PARA((i, e_t), Sol_4)$;
    $COST \longleftarrow Min(F(Sol_1), F(Sol_2), F(Sol_3))$;
    if $Tbest > COST$) **then**
      $Tbest \longleftarrow COST$;
      $TSol \longleftarrow X \in \{Sol_2, Sol_3, Sol_4\}$ with $F(X) = COST$
    end
  end
  $Sol \longleftarrow TSol$
end
Return $Tsol$;

5 COMPUTATIONAL RESULTS

The multi-start hybrid heuristic was coded in java with NetBeans IDE 7.0.1. Experiments were done on a windows seven PC of 2.53 GHz, Intel(R) Core(Tm) 2 Duo CPU with 3.49 GO of RAM. The validation method is structured into two phases: initially, the quality of the hybrid multi-start heuristic is validated on the PDP data sets without transshipment. Secondarily, the principle of transfer and gains it brings are studied through the application of the multi-start hybrid heuristic on a new PDPT data set. We have performed experiments on two types of instances: PDP and PDPT instances. The PDP data set was used to validate the multi-start heuristic without transshipment possibility. The PDPT data set was used to test all the hybrid multi-start heuristic. Multi-start heuristic is allowed to run for 20 minutes. The value of parameter λ was fixed to 4.

5.1 PDP data set results

To test the hybrid multi-start heuristic without transshipment (PDP), we used several instances from (Li and Lim, 2001). The instances are available on the site: www.top.sintef.no/vrp/benchmarks.html. This site lists the best results found by different authors on those instances. One can see (Li and Lim, 2001) for the detailed description of instances. Three classes of instances are considered: instances with random distributed customer (LR), instances with clustered customer (LC) and instances with partially and random distributed customers (LRC). Instances associated with number 1 have tight time windows; for example LC101. Instances associated with number 2 have large time windows. The results are summarized in the following table, where the columns are defined as follows: Name describes the name of instances, AK the average number of vehicles found with Multi-start, BK the best number of vehicles found with Multi-start, AD the average distance travelled by vehicles with Multi-start, BD the best travelled distance by vehicles found with Multi-start; WK the best result in term of vehicles, provided in the literature (best known results), WD the best result in term of distance provided in the literature. The informations on the best known solutions WK and WD are also given at the following address: www.top.sintef.no/vrp/benchmarks.htm.

Six solutions on twenty night tested do not give the best result, found by the authors having participated in the construction of the benchmark. The difference between the solutions we found and the best know solutions in literature remains however minimal be-

Name	BK	BD	WK	WD
lc101	**10**	**828,9**	**10**	**828,9**
lc102	**10**	**828,9**	**10**	**828,9**
lc103	**9**	1038.3	**9**	**1035,3**
lc104	**9**	864.1	**9**	**860,0**
lc105	**10**	**828,9**	**10**	**828,9**
lc106	**10**	**828,9**	**10**	**828,9**
lc107	**10**	**828,9**	**10**	**828,9**
lc108	**10**	**826,4**	**10**	**826,4**
lc109	**9**	1070.2	**9**	**1000,6**

Table 2: PDP: results

Name	BK	BD	WK	WD
lc201	**3**	**591,5**	**3**	**591,5**
lc202	**3**	**591,5**	**3**	**591,5**
lc203	**3**	591,2	**3**	**585,5**
lc204	**3**	**590.6**	**3**	**590,6**
lc205	**3**	**588,9**	**3**	**588,8**
lc206	**3**	**588,5**	**3**	**588,4**
lc207	**3**	**588,3**	**3**	**588,2**
lc208	**3**	**588,3**	**3**	**588,3**

Table 3: PDP: results

Name	BK	BD	WK	WD
lr101	**19**	**1650,8**	19	**1650,8**
lr102	**17**	**1487,5**	**17**	**1487,5**
lr103	**13**	**1292,6**	**13**	**1292,6**
lr104	**9**	**1013,3**	**9**	**1013,3**
lr105	**14**	**1377,1**	**14**	**1377,1**
lr106	**12**	**1252,6**	**12**	**1252,6**
lr107	**10**	**1111,3**	**10**	**1111,3**
lr108	**9**	**968.9**	**9**	**968,9**
lr109	**11**	**1208.9**	**11**	**1208,9**
lr110	11	1169	**10**	**1159,3**
lr111	**10**	**1108,9**	**10**	**1108,9**
lr112	10	1030.4	**9**	**1003,7**

Table 4: PDP: results

cause corresponding unless 1 %. For the twenty three other solutions, we were able to find the best know solution in a minimal time.

5.2 PDPT data set results

To validated the multi-start heuristic an analyse the advantage to practice transshipment, we have use new instances proposed by (Qu and Bard, 2012). For this experiment, the transshipment heuristic is used as post optimisation process to improve the overall PDP solutions found during all iteration. However, we cannot compare directly our results to those obtained by them because their working hypotheses are different from ours see (Qu and Bard, 2012). In the tables 2, we summarize the results obtained by the multi-start heuristic on different instances. We compare the pickup and delivery problem (PDP) against the pickup and delivery problem with transshipment (PDPT). The first two columns of the following tables are dedicated to PDP result and the two second one are dedicated to the PDPT. NV is the total number of vehicles used to satisfy customer request without transshipment, TD is the total travelled distances without transshipment, NVT is the total number of vehicles used with transshipment, and TDT is the total travelled distances with transshipment (post optimization).

Name	NV	TD	NVT	TDT
newdata401	**3**	**2835,08**	**3**	**2835,08**
newdata402	**3**	2841,30	**3**	**2823,23**
newdata403	**3**	2932,50	**3**	**2928,25**
newdata404	3	2846,42	**2**	**2405,80**
newdata405	**3**	**2822,81**	**3**	**2822,81**
newdata406	**3**	**2844,65**	**3**	**2844,65**
newdata407	**3**	**2908,20**	**3**	**2908,20**
newdata408	**3**	**2896,22**	**3**	**2896,22**
newdata409	**3**	**2890,18**	**3**	**2890,18**
newdata410	**3**	**2765,53**	**3**	**2765,53**

Table 5: PDPT: results

The analysis of all these results allows of us to show the positive impact of the transshipment on transportation. On practically all the instances, we have an improvement of the total travelled distance. At a time when the gas prices and energy costs become more and more expensive, this can be a source of relief for the finances of companies. For the instance newdata404 for example, the transshipment allows to reduce the number of used vehicles. Without transshipment the necessary number of vehicles is 3. With transshipment we only need 2 vehicles to satisfy the same number of requests. The consequence is immediate on the carbon footprint and on the costs of functioning of the company in terms of rent of vehicles. For the same instance, the transshipment reduces the total travelled distance by 16 %.

5.3 Conclusion

In this paper, we have introduced the principle of transshipment as a possible strategy to improve PDP solution. Due to the NP-hard characteristic of the problem, we have provide a multi-start hybrid heuristic based on variable neighbourhood descent and path relinking to tackle the pickup and delivery problem with and without transfer. The proposed heuristic was able to find almost altogether best know solution for the test problems. Will have also show the positive impact to practice transshipment. The continuation of this work will consist in studied the transshipment in a dynamic context.

ACKNOWLEDGMENTS

This research has been supported by ANR under the grant VTTPRODIGE (ANR-09-VTT0901) and is labelled by NOVALOG.

REFERENCES

Bent, R. and P.-V. Hentenryck, 2006. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers and Operations Research*, 33 (4), p. 875-893.

Boese, K.-D., A.-B. Kahng, and S. Muddu, 1994. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16 (2), p. 101-113.

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

Bräysy, O., G. Hasle, and W. Dullaert, 2004. A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research*, 159 (3), p. 586-605.

Brønmo, G., M. Christiansen, K. Fagerholt, and B. Nygreen, 2007. A multi-start local search heuristic for ship scheduling a computational study. *Computers and Operations Research*, 34 (3), p. 900-917.

Cortés, C.-E., M. Matamala, and C. Contardo, 2010. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200 (3), p. 711-724.

Crainic, T., S. Mancini, G. Perboli, and R. Tadei, 2011. *Multi-start Heuristics for the Two-Echelon Vehicle Routing Problem Evolutionary Computation in Combinatorial Optimization.* In: Merz, P. And Hao, J.-K. (eds.). Springer Berlin / Heidelberg.

Dumas, Y., J. Desrosiers, and F. Soumis, 1991. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54 (1), p. 7-22.

Hansen, P. and N. Mladenović, 2001. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130 (3), p. 449-467.

Ho, S. and M. Gendreau, 2006. Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12 (1), p. 55-72.

Li, H. and A. Lim, 2001. A metaheuristic for the pickup and delivery problem with time windows. In: Tools with Artificial Intelligence, *Proceedings of the 13th International Conference on, 7-9 Nov 2001.* p. 160-167.

Lu, Q. and M.-M. Dessouky, 2006. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175 (2), p. 672-687.

Savelsbergh, M.-W-.P., and M. Sol, 1995. The general pickup and delivery problem. *Transportation Science*, 29 (1), p. 17-29.

Martí, R., M. Laguna, and F. Glover, 2006. Principles of scatter search. *European Journal of Operational Research*, 169 (2), p. 359-372.

Martí, R., J.-M. Moreno-Vega, and A. Duarte, 2010. *Advanced Multi-start Methods*

Mitrovic-Minic, S. and G. Laporte, 2006. The pickup and delivery problem with time windows and transshipment. *In: INFOR, 2006.* p. 217-227.

Mues, C. and S. Pickl, 2005. Transshipment and Time Windows in Vehicle Routing. *In: ISPAN 2005, Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks*, p. 113-119.

Parragh, S., K. Doerner, and R. Hartl, 2008a. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58 1), p. 21-51.

Parragh, S., K. Doerner, and R. Hartl, 2008b. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58 (2), p. 81-117.

Qu, Y. and J.-F. Bard, 2012. A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers and Operations Research*, 39 (10), p. 2439-2456.

Resende, M. G. C., C.-C. Ribeiro, F. Glover, and R. Martí, 2010. *Scatter Search and Path-Relinking: Fundamentals, Advances, and Applications Handbook of Metaheuristics.* In: GENDREAU, M. and POTVIN, J.-Y. (eds.). Springer US.

Ropke, S. and D. Pisinger, 2006. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science* 40 (4), p. 455-472.

Shang, J. S. and C.-K. Cuff, 1996. Multicriteria pickup and delivery problem with transfer opportunity. *Computers and Industrial Engineering*, 30 (4), p. 631-645.