

# A Tabu Search Heuristic for the Pickup and Delivery Problem with Time Windows and a Fixed Size Fleet

Fabien Malca<sup>1</sup>, Frédéric Semet<sup>1</sup>

November 2003

<sup>1</sup>. LAMIH/ROI, UMR CNRS 8530, Université de Valenciennes et du Hainaut-Cambrésis, Le Mont Houy, 59313 VALENCIENNES Cedex 9.

**Keywords** : Pickup and Delivery Problem with Time Windows, Fixed size fleet, Tabu search heuristic.

In the *Pickup and Delivery Problem with Time Windows and a Fixed Size Fleet (m-PDPTW)*, a transportation carrier with a fixed size fleet of  $m$  vehicles receives a set of  $n$  requests. Each request consists of picking up a load, with a certain size, from some origin and to deliver it to a destination, in order to respect the time windows associated with the pickup location and the delivery location. Since the heterogenous fleet is finite, with finite capacity vehicles, some requests may be not assigned to a vehicle route without generating some delay on the time windows or exceeding the vehicle capacity. Then such requests cannot be accepted and are rejected. The aim is then to maximize the number of requests assigned to vehicles routes among the  $n$  requests, and next to minimize the total travel cost. Such a description scheme seems to be useful in the context of dynamic routing problems. The purpose of this paper is to provide a fast algorithm designed for the static case, before embedding it in a dynamic context in future work.

## 1 Notations and problem description

We begin this section by providing notations, then we define the problem studied.

### 1.1 Notations

We consider a set of  $n$  requests, all known in advance, denoted by  $O_j$  ( $j = 1, \dots, n$ ). For each request  $O_j$ ,  $O_j^+$  (resp.  $O_j^-$ ) is the pickup node (resp. the

delivery node),  $q_j$  is the size of the load to transport from  $O_j^+$  to  $O_j^-$ ,  $d_j^+$  (resp.  $d_j^-$ ) represents the pickup (resp. delivery) service duration time,  $[e_j^+, l_j^+]$  (resp.  $[e_j^-, l_j^-]$ ) is the time window in which pickup (resp. delivery) must start and finish. Note that if a vehicle arrives at any node before the beginning of the allowed time window, then it has to wait until the earliest possible arrival time.

The heterogeneous fleet has a fixed size of  $m$  vehicles. For each vehicle  $V^k$  ( $k = 1, \dots, m$ ),  $V_+^k$  (resp.  $V_-^k$ ) represents the starting depot (resp. ending) of the route  $R^k$  associated to vehicle  $V^k$ ,  $Q^k$  denotes the vehicle capacity, and  $[e_+^k, l_+^k]$  (resp.  $[e_-^k, l_-^k]$ ) is the time window in which  $V^k$  must leave the depot  $V_+^k$  (resp. the time window in which  $V^k$  must enter the depot  $V_-^k$ ).

We denote by  $c(A, B)$  (resp.  $t(A, B)$ ) the travel cost (resp. travel time) for edge  $(A, B)$  where  $A \neq B$  and  $A, B$  are in the set

$$\{O_j^+, O_j^-, V_+^k, V_-^k / j = 1, \dots, n; k = 1, \dots, m\}$$

Since we consider a fixed size fleet, all requests may not be assigned to a vehicle route. Thus we introduce a new route  $R^\infty$ , called *fictitious route*, in which all unassigned requests are placed. By contrast to the fictitious route, a vehicle route is also called a *real route*.

## 1.2 Problem description

The *m-PDPTW* consists of designing the route  $R^k$  for each vehicle  $V^k$  such that :

- (i) The route  $R^k$  starts at  $V_+^k$  and ends at  $V_-^k$  if the vehicle  $V^k$  serves one or several requests, otherwise, vehicle  $V^k$  stays in  $V_+^k$ .
- (ii) Every request belongs to exactly one route (the fictitious or a real one). For a request assigned to a real route, both pickup and delivery location belong to the same route, and pickup node is always visited before the corresponding delivery node.
- (iii) The total load of vehicle  $V^k$  never exceeds  $Q^k$ .
- (iv) Every node is visited during the allowed time windows.
- (v) The number of requests assigned to real routes is maximized and the total route cost is minimized.

Constraints (ii) are the coupling and precedence constraints, (iii) are capacity constraints and (iv) are time windows constraints.

We emphasize on the fact that we do not consider a biobjective problem, but a problem in which the objectives are aggregated in a hierarchical fashion. First,

we assign the maximum of requests, and then, we minimize the total route cost.

## 2 Main features of the tabu search algorithm

### 2.1 Solution space, cost and penalty functions, objective function

An important feature of our algorithm is that all constraints are relaxed (assignment, capacity and time windows). During the search we use the set  $S$  of solutions satisfying constraints (i) and (ii).

We introduce the following definitions to characterize different solution types :

- an *admissible* solution, is a solution satisfying all constraints for the requests assigned to real routes.
- a *feasible* solution, is a solution for which all requests are assigned to real routes and all constraints are satisfied.

For each solution  $s \in S$  visited during the search, we define the following functions :

- $n(s)$  : the number of requests in real routes
- $c(s)$  : the total route cost
- $q(s)$  : the total violation of capacity
- $w(s)$  : the total violation of time windows
- $c^\infty(s)$  : the penalty function used to represent the default in assignment of requests to real routes

We can also view the function  $c^\infty(s)$  as an evaluation of the fictitious route  $R^\infty$ . Its definition is  $c^\infty(s) = M \cdot (n - n(s))$ , where  $M$  is a constant term (a multiple of the most costly edge) used to adjust this penalty function according to the others in such a way that the hierarchy between the different terms is preserved.

Since we can consider unadmissible (and unfeasible) solutions during the search, instead of minimizing  $c^\infty(s)$  or  $c(s)$ , we attempt to minimize the *objective function*  $f(s)$  defined as a linear combination of all precedent functions

$$f(s) = c(s) + \alpha \cdot q(s) + \beta \cdot w(s) + \mu \cdot c^\infty(s)$$

where  $\alpha, \beta$  and  $\mu$  are positive parameters adjusted during the search to lead it in different areas of solution space.

## 2.2 Neighborhood structure

For any solution  $s$ , we define an *attribute set*  $B(s)$ , containing all attributes  $(R, O)$  such that request  $O$  belongs to the route  $R$  in solution  $s$ . Moves are then define on this attribute set basis, by removing and inserting an attribute.

Whenever we insert a request into any real route  $R^k$ , we have to insert both pickup and delivery locations. The number of inserting positions is of the order of the square of the number of nodes visited in the route. To minimize this number and then accelerate our algorithm, we propose to define an *elimination matrix set*,  $[E^k(O_i, O_j)]_{\substack{k=1, \dots, m \\ 1 \leq i, j \leq n}}$ , based on the possibility or not to combine two requests. This elimination matrix set can be computed in a preprocessing phase. This structure is then used in the definition of neighborhood structure to select only promising moves, by examining the compatibility of the candidate request for insertion with other requests of the route pairwise.

When the inserted request comes from a real route, this route is reconnected by linking predecessor to successor of both pickup and delivery.

## 2.3 Tabu status, aspiration criteria, diversification scheme

Based on the attribute set, we define the tabu status of a solution, its aspiration criteria and a diversification scheme.

Since the solution value may deteriorate during the search, we define tabu status on attributes, in order to avoid cycling. We define the matrix  $[\tau(R^k, O_j)]_{j=1, \dots, n}^{k=1, \dots, m, \infty}$  as follows : if we make at iteration  $\lambda$  the move  $O_j : R^k \rightarrow R^{k'}$ , where  $k \neq k'$ ;  $k, k' \in \{1, \dots, m, \infty\}$ , then, we declare tabu attribute  $(R^k, O_j)$  by setting  $\tau(R^k, O_j) = \lambda + \theta^1$ , where  $\theta^1$  is the main tabu tenure. Since insertion of  $O_j$  in  $R^{k'}$  is computed to be the best one, we declare also attributes  $(R^{k''}, O_j)$  tabu for all  $k'' \neq k, k'$  by setting  $\tau(R^{k''}, O_j) = \lambda + \theta^2$ , where  $\theta^2$  denotes the secondary tabu tenure, used to avoid moving the same request repeatedly. A tabu solution  $s'$  in  $N(s)$  is a solution such that the inserted attribute is tabu.

The tabu status for an admissible or feasible solution  $s'$  can be overridden. If we denote (and setup)  $\nu(R^{k'}, O_j)$  to be the best number of requests assigned in an admissible or feasible solution after moving request  $O_j$  to route  $R^{k'}$ , and  $\sigma(R^{k'}, O_j)$  the smallest route cost (with maximum number of requests assigned in real routes), then aspiration criteria is defined as :

- $n(s') > \nu(R^{k'}, O_j)$
- $n(s') = \nu(R^{k'}, O_j)$  and  $c(s') < \sigma(R^{k'}, O_j)$

To diversify the search, we add a penalty term to  $f(s')$  for each solution  $s' \in N(s)$  such that  $f(s') \geq f(s)$ . We use the same scheme used in [1] by counting the number of times that attribute  $(R^{k'}, O_j)$  has present in the current

solution; let us denote by  $\rho(R^{k'}, O_j)$  this value. The penalty term is then equal to

$$p(s') = \gamma \cdot \sqrt{(m+1) \cdot n} \cdot c(s') \cdot \sum_{(R,O) \in B(s')} \rho(R, O)$$

where  $\sqrt{(m+1) \cdot n} \cdot c(s')$  adjusts the penalties with the total solution cost and the problem size measured by the number of possible attributes.  $\gamma$  is a positive constant used to adjust the diversification intensity. By convention, if  $f(s') < f(s)$ , we set  $p(s') = 0$ .

## 2.4 Post-optimization heuristics, intensification

Since an request is inserted in a route without modifying relative positions of other locations in the route, we have adapted two heuristics proposed for the *TSP*, namely the *2-opt* and the *Or-opt*. These two heuristics are used as an intensification scheme, whenever a new local optima is reached.

## 3 Experimental results

Experimental results on benchmarks of Li and Lim<sup>1</sup> show that our method is fast (from 3 seconds to 2 minutes, for instances with 100 customers and about 10 vehicles) with a good quality of solution, and is promising for its adaptation to dynamic context.

## References

- [1] J.-F. Cordeau, G.Laporte, A. Mercier, "A Unified Tabu Search Heuristic for Vehicle Problem with Time Windows", *Journal of the Operational Research Society*, 52:928-936, 2001.
- [2] Z.Liang, H.C. Lau, "Pickup and Delivery with Time WIndows : Algorithms and Test Case Generation", *Working Paper (National University of Singapore)*, 2000.
- [3] H. Li, A. Lim, J. Huang, "Metaheuristic for solving the Pickup and Delivery Problem with Time Windows", *Working Paper (National University of Singapore)*, 2000.
- [4] W.P. Nanry, J.W. Barnes, "Solving the Pickup and Delivery Problem with Time Windows using Reactive Tabu Search", *Transportation Research : Part B*, 34:107-121, 2000.

---

<sup>1</sup><http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html>