Ant Colony Optimization Algorithm for Pickup and Delivery Problem with Time Windows

M. Noumbissi Tchoupo, A. Yalaoui, L. Amodeo, F. Yalaoui and F. Lutz

Abstract This paper presents an efficient meta-heuristic for the Pickup and Delivery Problem with Time Windows (PDPTW) based on Ant Colony Optimization coupled to dedicated local search algorithms. The objective function is the minimization of the number of vehicles and the minimization of the total distance travelled. In PDPTW, the demands are coupled and every couple is a request which must be satisfy in the same route. Thus, the feasible solution space is tightly constraint and then makes the design of effective heuristics more difficult. Experimental results on 56 instances of 100 customers of Li and Lim's benchmark show that the ACO coupled with PDPTW dedicated local search algorithms outperform existing algorithms. It returns in 98.2% (55/56) of cases a solution better or equal to the best known solution, and find a better solution than the best know in 44.6% (25/56).

1 Introduction

The Pickup and Delivery Problem with Time Windows (PDPTW) can be described as the design of a least cost routing plan to satisfy a set of transportation requests by a given identical vehicle fleet. Each request consists of delivering goods from a predefined location (pickup customer) to another one (delivery customer). In this

M.N. Tchoupo (✉) · A. Yalaoui (✉) · L. Amodeo (✉) · F. Yalaoui (✉)
University of Technology of Troyes, LOSI ICD UMR, CNRS, 6281 Troyes, France
e-mail: moise.noumbissitchoupo@utt.fr

A. Yalaoui
e-mail: alice.yalaoui@utt.fr

L. Amodeo
e-mail: lionel.amodeo@utt.fr

F. Yalaoui
e-mail: farouk.yalaoui@utt.fr

F. Lutz
Hospital of Troyes, 101 Av. Anatole France, Troyes, France
e-mail: frederic.lutz@hcs-sante.fr

A. Sforza and C. Sterle (eds.), *Optimization and Decision Science: Methodologies and Applications*, Springer Proceedings in Mathematics & Statistics 217,
DOI 10.1007/978-3-319-67308-0_19

problem, the routing plan is designed such that all vehicles start and end at the depot. The amount of goods must not exceed the vehicle's capacity. Each customer must be serviced within a given time windows. The service time indicates how long it will take for the pickup or delivery to be performed. For each request, the corresponding pickup customer must be visited before the corresponding delivery customer by the same vehicle and in the same route but not necessary immediately after. A vehicle is allowed to arrive at a location before the beginning of its time windows, and in this case must wait until the start of the time window. The problem is NP-hard as it contains the Travelling Salesman Problem with Time Windows (TSPTW) (See Dumas et al. [3]).

Numerous study have been done in PDPTW with the objective to minimize the number of vehicle (primary objective) and the total travelled distance (secondary objective). A simulated annealing with tabu search was proposed by Li and Lim [5] to solve PDPTW. Bent and Van Hentenryck [2] proposed a two-stage hybrid algorithm for PDPTW, where in the first stage the number of vehicles is decrease, while in the second stage the total travel cost is minimized by a Large Neighbourhood Search algorithm (LNS). An adaptive large neighbourhood search heuristic was proposed by Ropke and Pisinger [9]. Nagata and Kobayashi [6] successfully applied a Guided Ejection Search Algorithm to PDPTW. Nalepa et al. [7] proposed a parallel guided ejection search algorithm to solve PDPTW. In their approach, parallel processes co-operate periodically to enhance the quality of results and to accelerate the convergence of computations.

Tchoupo et al. [11] developed a Bender's decomposition algorithm for PDPTW with heterogeneous fleet (HVRPPDTW) to minimize the hierarchical objective. In the homogeneous case, their proposed approach was able to solve optimally instances up to 100 demands in reasonable computational time. To our knowledge, this method is the only exact algorithm for the PDPTW with hierarchical objective. For a survey on pickup and delivery problems see [8].

In the state of the art, there are not effective constructive heuristic to solve PDPTW. Indeed, the studies used iterative methods based on insertion and remove of requests. The greatest challenge in a constructive method is to find fast heuristics to choose the next demand to satisfy and verify there exists a path to achieve all delivery demands in current vehicle, whose corresponding pickup demands are not yet satisfied. Finding a feasible path to serve a given set of delivery demands is equivalent to solve a Hamiltonian path problem (NP-complete). This issue had been previously identified by Dumas et al. in [3] when they proposed a labelling algorithm for the Elementary Shortest Path Problem with Time Windows, Capacity, and Pickup and Delivery (ESPPTWCPD). In a proposed labelling algorithm, they proposed to consider only the subsets of deliveries of cardinality one and two.

The model proposed by Goss et al. [4] to explain the foraging behaviour of ants was the main source of inspiration for the development of ant colony optimization. The ACO was applied successfully to solve Vehicle Routing Problem as done by Belmecheri et al. [1] to solve the Vehicle Routing Problem with Heterogeneous fleet, Mixed Backhauls, and Time Windows.

To our knowledge, the proposed algorithm based on ACO algorithm coupled with local search algorithms depicted in this paper is the first effective constructive algorithm for the addressed problem in this study.

The remainder of the paper is organized as follow. Section 2 proposes a mixed integer linear program for the PDPTW problem. Section 3 describes the ACO, with the pheromone initialization, their updating and the computation of the visibility. Section 4 presents three local search algorithms. The setting of parameters and the experimental results are reported in Sect. 5.

2 Mathematical Model

This section presents a new mixed integer linear program to model the addressed problem. It is based on the model proposed by [11] for the PDPTW with heterogeneous fleet. We note N the set of $2n$ customers, node 0 represents depot (origin and destination) and $V = N \cup \{0\}$ the set of $2n + 1$ nodes. $P = \{1, \cdots, n\}$ is the set of pickup demands, the set of n delivery demands is noted $D = \{n + 1, \cdots, 2n\}$, K is the set of m identical vehicles with a capacity of Q items. A is the set of arcs, δ is the fixed cost of using a vehicle, d_{ij} represents the distance between the vertices i and j, t_{ij} is the time between the location of vertices i and j, s_i represents the service time required by the node i, $|q_i|$ is the amount of goods to pickup or delivery, e_i the earlier time at which the service may begin at node i and l_i the latest time at which the service may begin at node i. We assume that:

$$\forall i \in P, \quad q_i > 0 \quad \text{and} \quad q_{n+i} = -q_i \quad \text{and} \quad (i,j) \in A \Longleftrightarrow e_i + s_i + t_{ij} \leq l_j.$$

The problem is formulated as a mixed integer linear program (MILP). For each arc $(i,j) \in A$ and each vehicle $k \in K$, let x_{ij}^k be a binary variable equals to 1 if the vehicle k travels from location of demand i to location of demand j, and 0 otherwise. For each node $i \in N$, and each vehicle $k \in K$, let B_i^k the time at which vehicle k begins the service at node i. Q_{ij}^k is the load of vehicle k on the arc (i,j). The formulation is the following:

$$Min \quad \sum_{i \in P} \delta x_{0i}^k + \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij}^k \tag{1}$$

$$\sum_{(i,j) \in A} \sum_{k \in K} x_{ij}^k = 1, \qquad \forall i \in P; \tag{2}$$

$$\sum_{j|(i,j) \in A} x_{ij}^k = \sum_{j|(n+i,j) \in A} x_{n+i,j}^k, \qquad \forall (i,k) \in P \times K; \tag{3}$$

$$\sum_{i \in P} x_{0i}^k \leq 1, \qquad \forall k \in K; \tag{4}$$

$$\sum_{i\in P} x_{0i}^k = \sum_{i\in D} x_{i0}^k, \qquad \forall k \in K; \tag{5}$$

$$\sum_{j|(i,j)\in A} x_{ij}^k = \sum_{j|(j,i)\in A} x_{ji}^k, \qquad \forall (i,k) \in P \cup D \times K; \tag{6}$$

$$\sum_{j|(i,j)\in A} x_{ij}^k = \sum_{i\in P} x_{0i}^k, \qquad \forall (i,k) \in P \cup D \times K; \tag{7}$$

$$\sum_{j|(i,j)\in A} \sum_{k\in K} Q_{ij}^k + \sum_{j|(j,i)\in A} \sum_{k\in K} Q_{ji}^k = q_i, \quad \forall i \in P; \tag{8}$$

$$\sum_{i\in P} \sum_{k\in K} Q_{0i}^k + \sum_{i\in D} \sum_{k\in K} Q_{i0}^k = 0; \tag{9}$$

$$Q_{ij}^k \le Q \times x_{ij}^k, \qquad \forall (i,j) \in A,\ \forall k \in K; \tag{10}$$

$$B_i^k - l_i + (l_i + s_i + t_{ij}) x_{ij}^k \le B_j^k, \qquad \forall (i,j) \in A,\ \forall k \in K; \tag{11}$$

$$e_i \sum_{j|(i,j)\in A} x_{ij}^k \le B_i^k \le l_i \sum_{j|(i,j)\in A} x_{ij}^k, \qquad \forall (i,k) \in N \times K; \tag{12}$$

$$B_i^k + (s_i + t_{i,n+i}) \times \sum_{j\in N} x_{ij}^k \le B_{n+i}^k, \qquad \forall (i,k) \in P \times K; \tag{13}$$

$$x_{ij}^k \in \{0,1\},\ Q_{ij}^k \ge 0,\ B_i^k \ge 0, \qquad \forall (i,j) \in A,\ \forall k \in K. \tag{14}$$

The objective function (1) minimizes the number of vehicles used and the total distance travelled. Constraints (2) and (3) ensure that each pickup demand is served exactly once and the corresponding delivery demand is served by the same vehicle in the same route. Constraints (4) and (7) guarantee that the route of each vehicle starts and ends at the depot. The respect of the time windows and the capacity of vehicle is ensured by constraints (8) to (12). Constraint (13) assures that every delivery demand is satisfied after the corresponding pickup demand but not necessary immediately after the pickup point.

3 Ant Colony Optimization (ACO)

In this study, we apply a variant of ACO called Ant Colony System (ACS), characterized by introduction of a local pheromone update. Ant colony optimization is chosen because it is a constructive method which does not require reparation procedure.

3.1 Construction of Solution

A solution is composed of a set of routes and each route is realized by one vehicle. An ant constructs the routes of a solution sequentially. Ant keeps inserting demands in the current route as long as the are non-satisfied demands that respect the constraints (capacity, times windows and paring). If in a partial solution there still are non-visited nodes but none of them can be inserted in the route being built, the ant close the current route and start a new one. Let ρ a partial solution formed by $r-1$ complete routes and a r^{th} route in construction. Let i the last demand completed in route r, Q_r the load of the vehicle after satisfied demand i and $\mathcal{V}$ the set of unsatisfied delivery demands such that their corresponding pickup demands has been served. A demand j is *eligible* to be satisfy if it didn't have been completed yet and if one of these conditions is satisfied:

1. $0 < j \leq n$ and there exists a path to satisfied all demands in $\mathcal{V} \cup \{n+j\}$
2. $n < j \leq 2n \wedge j - n \in r$ and there exists a path to satisfied all demands in $\mathcal{V} \setminus \{j\}$

An eligible demand j to be inserted in the partial solution ρ is chosen randomly using probability:

$$P_{ij}^{\rho} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij}^{\rho})^{\beta}}{\sum_{d \in S_i^{\rho}} (\tau_{ij})^{\alpha}(\eta_{ij}^{\rho})^{\beta}} \text{ if } j \in S_i^{\rho}, \text{ and 0 otherwise.} \tag{15}$$

P_{ij}^{ρ} represents the probability to choose a demand j to complete from the current demand i. τ_{ij} denotes the trail of pheromone on arc (i,j). S_i^{ρ} is the set of eligible demands that we can performed after demand i. The parameters α et β modulate the importance between the visibility and the pheromone. η_{ij}^{ρ} is the visibility value used to guide ant.

$$\eta_{ij}^{\rho} = \frac{\alpha_1 |S_j^{\rho \cup \{j\}}|}{d_{ij}} \tag{16}$$

with $\alpha_1 \in]0, 1]$ a fixed scalar.

Given a partial solution ρ, ending by satisfying demand i, the eligibility of a demand j is obtained by finding a feasible path in a graph. Indeed, it shall be demonstrate that after performed demand j, there exists a feasible path to satisfy all unsatisfied delivery demands whose corresponding pickup demands were satisfied in route being built in ρ. This problem is a Hamiltonian path problem, which is NP-complete and to solve it, an insertion heuristic Algorithm 1 is proposed.

3.2 Pheromone Updating

At the beginning of ACS, pheromones are initialized by:

Algorithm 1 Insertion heuristic for Hamiltonian path problem

1: Inputs: $\mathcal{V}$ (a set of demand to satisfy),*ItMax*1 (a number of iterations), i (the current demand) and t (the time at the end of service of demand i)
2: **while** number of iterations < *ItMax*1 and no feasible path which satisfy all demands in $\mathcal{V}$ is found **do**
3: $\mathcal{V}' = \mathcal{V}$
4: Initialize a route r beginning at node i at the time t
5: **while** $\mathcal{V}'$ in not empty **do**
6: Choose a random delivery $d \in \mathcal{V}'$, remove d in $\mathcal{V}'$, and try to insert it in r at the best position.
7: **if** d is not inserted in r **then**
8: Go back to step 3.
9: **end if**
10: **end while**
11: **end while**

$$\tau_{ij} = \tau_0 \text{ if } (i,j) \in A, \text{ and 0 otherwise.} \tag{17}$$

with τ_0 a fixed scalar. As we mentioned, ACS has two types of pheromone update:

- local updating : $\tau_{ij} = \epsilon_1 \tau_{ij} + \tau_0$,

used to diversify the search in a given iteration.

- global updating used:

$$\tau_{ij} = \epsilon_2 \tau_{ij} + (1 - \epsilon_2)\Delta^*_{ij} \text{ if } (i,j) \text{ belong in the best ant, and } \epsilon_2 \tau_{ij} \text{ otherwise.} \tag{18}$$

with $\Delta^*_{ij} = \frac{(\text{number of demands in } r^*_i)^{\alpha_2}}{\text{total distance travelled on } r^*_i}$, r^*_i the route which contain the demand i, $\epsilon_1, \epsilon_2 \in]0,1[$ fixed and α_2 a positive fixed scalar.

4 Local Search Algorithm

This section describes three local search algorithms used in this work. Each local search is dedicated to specific feature of the objective function of PDPTW.

Heuristic **H1** is inspired from the two-stage method used in [6]. For a given solution, H1 is used to decrease the number of vehicles. For a each route in the solution, H1 removes a route from it, and try to insert all its requests in the remaining routes of the solution. The order of insertion is the order of completion in the delete route. Every request is inserted in the route and in the positions (pickup and delivery positions) that minimize the total distance travelled. If finally, all the request presents in the deleted route are inserted, the solution is updated.

The second heuristic **H2** is proposed to reduce distance travelled for a given route. The idea is to generate randomly (uniform distribution) an insertion order of pickup demand. And inserted at the best position the requests in this order.

The third local search **H3** is proposed to minimize total distance travelled for a given solution. At each iteration, a route r and a demand d (in r) are chosen randomly. The corresponding couple of pickup and delivery demands for d is remove from the solution and reinserted in the route and at the position whom minimise total cost. H3 is inspired from a part of LNS algorithm developed by S. Ropke in [8]. The pseudo code of hybrid ACS used is given by Algorithm 2.

Algorithm 2 Pseudo code of hybrid ACS

```
Initialization of parameters
while the best solution is not improved in ItMax iterations do
  for each ant of the population do
    Construct a solution to complete all demands
    while we can remove a route do
      Apply respectively algorithms: H2, H1 and H3
    end while
    if the current solution have the same number of vehicles as the best solution found then
      while we decrease the total distance travelled do
        Apply respectively algorithms: H2 and H3
      end while
    end if
    Apply the local updating pheromone
  end for
  Apply the global updating pheromone
end while
```

In the algorithm, the order H2, H1 and then H3 is used to first optimize each route, and then try to decrease the vehicles number and finally, minimize the total distance travelled.

5 Computational Results

The proposed approach has been implemented on eclipse, the programming language was C++ and the experiments have been carried on a 1.5 GHz and 3.3 GB of RAM. Standard Li et Lim's benchmark [9] is chosen to evaluate the performance of our approach. For experiments, we keep the same value $\epsilon = 0.9$ proposed by Belmecheri et al. [2]. A sensibility analysis is made in order to fix the following parameters: Number of ants $\in \{\frac{n}{2}, \frac{2n}{5}\ \frac{n}{3}\}$, $\alpha \in \{2, 3, 4\}$, $\beta \in \{1, 2\}$, $\alpha_1 \in \{0.1, 0.2, 0.5, 1\}$, $\alpha_2 \in \{2, 3, 5, 10\}$, $\tau_0 \in \{0.01, 1, 10\}$, $ItMax1 \in \{5, 10, 20\}$ and $Itmax2 \in \{n, 2n, 3n\}$. We obtain with this analysis that the best values are : Number of ants $= \frac{2n}{5}$, $\alpha = 3$, $\beta = 1$, $\alpha_1 = 0.2$, $\alpha_2 = 5$, $\tau_0 = 1$, $ItMax1 = 10$, $Itmax2 = 2n$. The algorithm stops when the best solution is not improved after 100 iterations.

Table 1 Experiments of ACO algorithm

Best known solution				Hybrid ACS			Best known solution				Hybrid ACS		
Instance	NV	TD	REF	NV	TD	CPU	Instance	NV	TD	REF	NV	TD	CPU
lc101*	10	828,94	[11]	10	828,94	86	lr112	9	1003,77	[5]	**9**	**1002,38**	167
lc102	10	828,94	[5]	10	828,94	103	lr201	4	1253,23	[9]	4	1253,23	207
lc103	9	1035,35	[2]	**9**	**968,92**	96	lr202	3	1197,67	[5]	3	1197,67	455
lc104	9	860,01	[9]	9	860,01	156	lr203	3	949,4	[5]	3	949,4	556
lc105	10	828,94	[5]	10	828,94	37	lr204	3	849,05	[5]	**3**	**847,83**	906
lc106	10	828,94	[5]	10	828,94	43	lr205	3	1054,02	[5]	**3**	**1053,98**	134
lc107	10	828,94	[5]	10	828,94	39	lr206	3	931,63	[5]	3	931,63	641
lc108	10	826,44	[5]	10	826,44	49	lr207	2	903,06	[5]	**2**	**902,24**	434
lc109	9	1000,60	[2]	10	827,82	96	lr208	2	734,85	[5]	**2**	**734,09**	1555
lc201*	3	591,56	[11]	3	591,56	49	lr209	3	930,59	[9]	3	930,59	234
lc202*	3	591,56	[11]	3	591,56	260	lr210	3	964,22	[5]	3	964,22	375
lc203	3	591,17	[9]	3	591,17	190	lr211	2	911,52	[9]	**2**	**903,76**	1525
lc204	3	590,6	[9]	**3**	**590,39**	634	lrc101	14	1708,8	[5]	14	1708,77	98

(continued)

Table 1 (continued)

Best known solution				Hybrid ACS				Best known solution				Hybrid ACS		
lc205*	3	588,88	[11]	3	588,88	77		lrc102	12	1558,07	[9]	**12**	**1556,82**	49
lc206	3	588,49	[5]	3	588,49	144		lrc103	11	1258,74	[5]	**11**	**1256,06**	57
lc207	3	588,29	[5]	3	588,29	102		lrc104	10	1128,40	[5]	**10**	**1126,23**	98
lc208	3	588,32	[5]	3	588,32	108		lrc105	13	1637,62	[5]	**13**	**1633,56**	52
lr101	19	1650,80	[5]	19	1650,8	31		lrc106	11	1424,73	[9]	11	1424,73	210
lr102	17	1487,57	[5]	**17**	**1487,49**	54		lrc107	11	1230,14	[9]	**11**	**1225,68**	58
lr103	13	1292,68	[5]	**13**	**1284,93**	61		lrc108	10	1147,43	[9]	10	1147,96	70
lr104	9	1013,39	[5]	**9**	**999,27**	87		lrc201	4	1406,94	[9]	**4**	**1396,88**	195
lr105	14	1377,11	[5]	14	1377,11	31		lrc202	3	1374,27	[5]	**3**	**1361,24**	153
lr106	12	1252,62	[5]	**12**	**1248,93**	37		lrc203	3	1089,07	[5]	3	1089,07	350
lr107	10	1111,31	[5]	**10**	**1101,89**	59		lrc204	3	818,66	[9]	3	818,66	885
lr108	9	968,97	[5]	**9**	**966,30**	59		lrc205	4	1302,20	[5]	4	1302,20	274
lr109	11	1208,96	[9]	**11**	**1208,92**	43		lrc206	3	1159,03	[9]	**3**	**1156,54**	135
lr110	10	1159,35	[5]	**10**	**1158,27**	113		lrc207	3	1062,05	[9]	**3**	**1054,24**	205
lr111	10	1108,9	[5]	10	1108,9	74		lrc208	3	852,76	[5]	3	852,76	469

Table 1 contains five types of columns: *Instance* is the name of instance, *NV* is the number of used vehicles, *TD* is the total travelled distance, *REF* is a reference to the paper which found the result and *CPU* is the computational time in seconds. The symbol "$*$" indicates that the instance is solved to optimality and the values in bold emphasize that the proposed algorithm found a better solution.

The hybrid ACS algorithm returns in 98.2% (55/56) of cases a solution better or equal to the best known solution. And find a better solution than the best know solution in 44.6% (25/56). It is important to remark that our method is able to performed best known solutions on all configurations: lc (clustered), lr (Uniform distributed) and lrc (Semi-clustered).

6 Conclusion

This paper proposes an efficient algorithm to solve a Pickup and Delivery Problem with Time Windows with objective function : first minimize the number of vehicles and second minimize the total distance travelled. The proposes algorithm is based on ant colony optimization coupled with three fast local search algorithms. To our knowledge, this approach is the first constructive method to solve PDPTW problem with this objective. The experiments on the standard PDPTW benchmark of Li and Lim [10] show that it outperforms existing algorithms. In the future, it would be interesting to performed every feature of approach (construction, visibility, update of pheromone and local search algorithms) to solve more large size instances and generalized this approach on heterogeneous fleet case.

References

1. Belmecheri, F., et al.: An ant colony optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. In: 13th IFAC Symposium on Information Control Problems in Manufacturing. Moscow, Russia (2009)
2. Bent R, van Hentenryck P.: A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. In: Rossi, F. (ed.) Principles and Practice of Constraints Programming, Springer, Heidelberg, Berlin (2003)
3. Dumas, Y., Desrosiers, J.: The pickup and delivery problem with time windows. Eur. J. Operation. Res. **54**, 7–22 (1991)
4. Goss, S., Aron, S., Deneubourg, J., Pasteels, J.: Self-organized shortcuts in the Argentine ant. Naturwissenschaften **76**, 579–581 (1989)
5. Li, A., Lim, H.: A metaheuristic for the pickup and delivery vehicle routing problems with time windows. In: IEEE Computer Society, 13th IEE International Conference on Tools with Artificial Intelligence (ICTAI-01), pp. 333–340. Los Alamitos, USA (2001)
6. Nagata, Y., Kobayashi.: Guided ejection Search for the pickup and delivery problem with time windows. In: Cowling, P., Merz, P. (eds.) Evolutionary Computation in Combinatorial Optimization, Springer, Berlin, Heidelberg (2010)

7. Nalepa, J., Blocho, M.: A parallel algorithm with the search space partition for the pickup and delivery with time windows. In: Proceedings of 10th International Conference on P2P, Parallel, Grid, Cloud and internet Computing (IEE 3PGCIC), pp. 92–99 (2015)
8. Parragh, S., Doerner, K., Hartl, R.: A survey on pickup and delivery problems. J. fur Betriebswirtschaft **58**(2), 81–117 (2008)
9. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Trans. Sci. **40**(4), 455–472 (2006)
10. SINTEF vehicle routing and traveling salesperson problems. https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmrak/100-customers/
11. Tchoupo, M.N., Yalaoui, A., Amodeo, L., Yaloui, F., Lutz, F.: Problème de collectes et livraisons avec fenêtres de temps et flotte hétérogène. In: 11th International Conference on Modeling, Optimization & Simulation. Montreal, Canada (2016)