A particle swarm optimization for selective pickup and delivery problem

1st Zhihao PENG
Univ. Bourgogne Franche-Comté FEMTO-ST Institute/CNRS,
Rue Thierry-Mieg (UTBM),
90 010 Belfort Cedex, France
zhihao.peng@utbm.fr

2nd Zaher Al Chami
Univ. Bourgogne Franche-Comté FEMTO-ST Institute/CNRS,
Rue Thierry-Mieg (UTBM),
90 010 Belfort Cedex, France
zaher.al-chami@utbm.fr

3rd Hervé Manier
Univ. Bourgogne Franche-Comté FEMTO-ST Institute/CNRS,
Rue Thierry-Mieg (UTBM),
90 010 Belfort Cedex, France
herve.manier@utbm.fr

4th Marie-Ange Manier
Univ. Bourgogne Franche-Comté FEMTO-ST Institute/CNRS,
Rue Thierry-Mieg (UTBM),
90 010 Belfort Cedex, France
marie-ange.manier@utbm.fr

Abstract—This paper studies a variant of vehicle routing problem called Selective Pickup and Delivery Problem with Time Windows and Paired Demands (SPDPTWPD). A visiting sequence of each assigned vehicle needs to be determined by respecting the imposed constraints. Like for other combinatorial problems, the optimal solution cannot be obtained in a reasonable time when the size increases. An approached method is thus chosen as an alternative to tackle this issue. The proposed method integrates particle swarm optimization (PSO) with local searches by considering the diversification of PSO and intensification of local search. To validate the method, experiments are made on the benchmarks from the literature. The experiments are divided into two parts. In the first part, a self-comparison is made to demonstrate the evolutionary capacity of PSO and the efficiency of proposed local searches. In the second part, the proposed method is compared with a genetic algorithm from the literature. The results show that the method is competitive and efficient.

Index Terms—PSO, Pickup and delivery problem, Meta-heuristic

I. INTRODUCTION

The pickup and delivery problem (PDP) studies the problem in which all the demands should be transported from pickup points (suppliers) to delivery points (customers) by vehicles. One of the most studied contexts of PDP is the urban area where the capacity of vehicle and the visiting time are limited. In the literature, this problem is called pickup and delivery problem with time windows (PDPTW).

In this paper, a variant is studied called selective pickup and delivery problem with time windows and paired demand (SPDPTWPD). The selective variant relaxes the constraint in which all the demands should be satisfied, and the paired demand variant associates one supplier with its related customer. To better explain the problem, a sketch has been presented in Fig.1. In the figure, there are four paired demands which are identified by the same color. The first vehicle departs from the depot, and visits the supplier of the first demand. Then it visits the customer of the first demand, and returns to the depot. The second vehicle departs from the depot, and visits suppliers of demand two and three sequentially. Then it visits related customers, and returns to the depot. The demand four is not satisfied by any vehicles. When a vehicle is making the route, several constraints should be respected: 1) It departs and returns to the same depot. 2) The carried demands should not exceed its maximum capacity. 3) When it arrives earlier than the opening time of a site (earliest arrival time), it should wait until this site is available. It should never arrive later than the close time of a site (latest arrival time). Since the studied problem is a generalization of the well-known vehicle routing problem (VRP), it is a NP-hard problem.

Several applications can be found in the real world. For example, if the resource (vehicles) of a company is limited, as consequence it may not satisfy all the demands. In this case, the decision maker can associate each demand with a profit. Maximizing the profit (satisfying the most important demands) is then considered as the first objective. A solution of this problem is a set of visiting sequence of each assigned vehicle. Since there may be several solutions corresponding to the first objective, minimizing the distance can be considered as the second objective to choose the most economical solution.

Fig. 1. Example of SPDPTWPD.
II. LITERATURE REVIEW

In this section, related work about SPDPTWPD and PSO will be presented briefly.

The VRP is a problem in which a set of vehicles needs to find the best routes to visit all the customers. Many contributions have been made regarding this problem or its extensions. To get a better understanding, please refer to the recent survey [1].

In PDP, integrating pickup and delivery operations in the same route makes the problem more complicated. Extensive surveys are presented in [2], [3], [4]. Among the variants of PDP, PDPTW is the most studied one. As to exact methods, a new branch-and-cut-and-price algorithm was introduced in [5] by using column generation. An elementary and a non-elementary shortest path problem was considered as two pricing subproblems. In [6], an exact method based on a set-partitioning–like integer formulation was proposed. As to heuristic methods, a method integrating Squeaky Wheel Optimization, Solomon’s Insertion Heuristic, and Local Search has been introduced in [7]. Minimizing the fleet size, traveling distance, scheduling durations and waiting times are considered as objectives. In [8], a tabu-embedded simulated annealing algorithm was proposed to solve large multiple-vehicle PDPTW instances with various distribution properties.

The SPDPTWPD was firstly studied in [9], in which an arc-based mixed linear program has been introduced. Several additional constraints were proposed to accelerate the solving of the model. Based on the instances of Li and Lim [8], new instances were generated to validate the model. In [10], the same problem was studied by taking into account two objectives: maximizing the profit and minimizing the distance. To deal with two objectives, an lexicographic approach was proposed by respecting a predefined execution sequence. Although two objectives were considered in this paper, only one non dominated solution was obtained for each instance instead of a complete pareto front. Due to the complexity of the problem, a metaheuristic based on a combination between tabu search and simulated annealing was introduced in [11]. The drawback of the proposed method is that only one non dominated solution can be obtained. Therefore, hybrid genetic algorithm (HG) was proposed in [12]. The method can find high quality solutions, but the number of non dominated solutions was also limited.

As a nature inspired algorithm, PSO was originally designed for solving continuous problems. To adapt PSO to discrete problems, a random key (RK) technique is often used. In [13], PSO combined with local search was used to solve the capacitated location routing problem. In [14], two solution representations were proposed to solve a variant of PDPTW. PSO can also be used to solve multi-objective problems. In [15], MOPSO combined with adapted multi-objective variable neighborhood search (AMOVNS) was used to solve a multi-echelon vehicle routing problem, in which delivery procedure is divided into different stages, each stage is called one echelon.

The SPDPTWPD has been little studied. To our best knowledge, only two metaheuristics are proposed in [12] and in [11] to solve the problem. In this paper, a hybrid multi-objective PSO is proposed to solve this variant.

III. PROBLEM STATEMENT

In this section, a description of SPDPTWPD is given. A set of \( n \) demands, \( R = \{1,...,n\}, \) is considered. Each demand \( r \in R \) is associated with a pickup point \( p(r) \) and a delivery point \( d(r) \). The set of all pickup and delivery points are denoted by \( P \) and \( D \). A profit \( \text{profit}_r \) can be gained if demand \( r \) with quantity \( q_r \) is satisfied. \( s_r \) is the loading/unloading time.

In this section, related work about SPDPTWPD and PSO will be presented briefly.

The VRP is a problem in which a set of vehicles needs to find the best routes to visit all the customers. Many contributions have been made regarding this problem or its extensions. To get a better understanding, please refer to the recent survey [1].

In PDP, integrating pickup and delivery operations in the same route makes the problem more complicated. Extensive surveys are presented in [2], [3], [4]. Among the variants of PDP, PDPTW is the most studied one. As to exact methods, a new branch-and-cut-and-price algorithm was introduced in [5] by using column generation. An elementary and a non-elementary shortest path problem was considered as two pricing subproblems. In [6], an exact method based on a set-partitioning–like integer formulation was proposed. As to heuristic methods, a method integrating Squeaky Wheel Optimization, Solomon’s Insertion Heuristic, and Local Search has been introduced in [7]. Minimizing the fleet size, traveling distance, scheduling durations and waiting times are considered as objectives. In [8], a tabu-embedded simulated annealing algorithm was proposed to solve large multiple-vehicle PDPTW instances with various distribution properties.

The SPDPTWPD was firstly studied in [9], in which an arc-based mixed linear program has been introduced. Several additional constraints were proposed to accelerate the solving of the model. Based on the instances of Li and Lim [8], new instances were generated to validate the model. In [10], the same problem was studied by taking into account two objectives: maximizing the profit and minimizing the distance. To deal with two objectives, an lexicographic approach was proposed by respecting a predefined execution sequence. Although two objectives were considered in this paper, only one non dominated solution was obtained for each instance instead of a complete pareto front. Due to the complexity of the problem, a metaheuristic based on a combination between tabu search and simulated annealing was introduced in [11]. The drawback of the proposed method is that only one non dominated solution can be obtained. Therefore, hybrid genetic algorithm (HG) was proposed in [12]. The method can find high quality solutions, but the number of non dominated solutions was also limited.

As a nature inspired algorithm, PSO was originally designed for solving continuous problems. To adapt PSO to discrete problems, a random key (RK) technique is often used. In [13], PSO combined with local search was used to solve the capacitated location routing problem. In [14], two solution representations were proposed to solve a variant of PDPTW. PSO can also be used to solve multi-objective problems. In [15], MOPSO combined with adapted multi-objective variable neighborhood search (AMOVNS) was used to solve a multi-echelon vehicle routing problem, in which delivery procedure is divided into different stages, each stage is called one echelon.

The SPDPTWPD has been little studied. To our best knowledge, only two metaheuristics are proposed in [12] and in [11] to solve the problem. In this paper, a hybrid multi-objective PSO is proposed to solve this variant.

In this section, a description of SPDPTWPD is given. A set of \( n \) demands, \( R = \{1,...,n\}, \) is considered. Each demand \( r \in R \) is associated with a pickup point \( p(r) \) and a delivery point \( d(r) \). The set of all pickup and delivery points are denoted by \( P \) and \( D \). A profit \( \text{profit}_r \) can be gained if demand \( r \) with quantity \( q_r \) is satisfied. \( s_r \) is the loading/unloading time.

In this section, a description of SPDPTWPD is given. A set of \( n \) demands, \( R = \{1,...,n\}, \) is considered. Each demand \( r \in R \) is associated with a pickup point \( p(r) \) and a delivery point \( d(r) \). The set of all pickup and delivery points are denoted by \( P \) and \( D \). A profit \( \text{profit}_r \) can be gained if demand \( r \) with quantity \( q_r \) is satisfied. \( s_r \) is the loading/unloading time.

In this section, a description of SPDPTWPD is given. A set of \( n \) demands, \( R = \{1,...,n\}, \) is considered. Each demand \( r \in R \) is associated with a pickup point \( p(r) \) and a delivery point \( d(r) \). The set of all pickup and delivery points are denoted by \( P \) and \( D \). A profit \( \text{profit}_r \) can be gained if demand \( r \) with quantity \( q_r \) is satisfied. \( s_r \) is the loading/unloading time.

In this section, a description of SPDPTWPD is given. A set of \( n \) demands, \( R = \{1,...,n\}, \) is considered. Each demand \( r \in R \) is associated with a pickup point \( p(r) \) and a delivery point \( d(r) \). The set of all pickup and delivery points are denoted by \( P \) and \( D \). A profit \( \text{profit}_r \) can be gained if demand \( r \) with quantity \( q_r \) is satisfied. \( s_r \) is the loading/unloading time.
the best solution ever found by the group (Gbest). The evolution of each particle is affected by its current solution, Pbest and Gbest. This mechanic can well explain that PSO imitates the social activity from the nature. As PSO performs well in single-objective problems, researchers tried to adapt it to multi-objective problems [16]. In MOPSO, a strategy needs to be proposed to select a leader from a set of non dominated solutions. Different variants have been proposed in the literature. Since [17] is one of the first papers trying to solve multiobjectives by using PSO, it’s easier to adapt it to multi-objective problems [16]. In MOPSO, a strategy to remove several solutions. However, we consider the capacity (capacity) is reached, a strategy should be proposed to remove several solutions. If the limited number of dominated solutions are removed. If the limited number of solutions (capacity) is reached, a strategy should be proposed to remove several solutions. However, we consider the capacity is unlimited. As to the selection of a leader, two strategies are studied in this paper: First-In-First-Out (FIFO), fitness-based on sharing method is employed for REP_GBEST.

B. Encoding and Decoding

A good solution representation is important when dealing with combination optimization problem. A solution representation is shown in Fig.3.

![Fig. 3. Representation of solution](image)

In the figure, a particle is divided into two parts. In the first part, each demand in [1…r] is assigned with two dimensions. The value of each dimension is randomly generated in [0…10]. By sorting the value in ascending order, a priority list (visiting sequence A) can be generated. In the list, the first dimension of a demand is the pickup point, the second dimension of a demand is then the delivery point. This is shown in Fig.4.

![Fig. 4. Visiting sequence A](image)

In the second part, each demand is assigned with one dimension. The value of each dimension is randomly generated in [0,…, NV], where NV is the maximum number of vehicles. A floor operation is performed on each value to get the index of assigned vehicle. This is shown in Fig.5.

By using the information provided above, Algorithm1 defines the construction of tours. In the algorithm, the Insert method adds the pair (supplier, customer) into vehicle of
S at the position which is defined by visiting sequence A. The output is a complete solution S, and a set of unsatisfied demands D.

**Algorithm 1: Construction of tours**

**Input:** Visiting sequence A, Vehicle assignment B;  
**Output:** Solution S, Demands not satisfied D;  
D = ∅;  
for each element i in A do  
  if i is a supplier then  
    demand = GetDemand(i);  
    supplier = i;  
    customer = GetCustomer(demand);  
    vehicle = GetAssignedVehicle(B, demand);  
    Insert(S, vehicle, supplier, customer, A);  
  if S is not feasible then  
    Remove(S, vehicle, supplier, customer);  
    D.add(demand);  
  end  
end

**C. Local searches**

There are mainly two defaults of the proposed solution representation: 1) when NV increases, a vehicle may satisfy only one demand during its tour in certain cases. As we believe that traveling distance will be increased if vehicles are not charged. 2) Unsatisfied demands are not treated. Although this is allowed from the selective aspect, the profit achieved by the population will be possibly very weak. Two local searches LS1 and LS2 are thus proposed to cover the two defaults. For LS1, it plays two roles: Firstly, it tries to minimize the number of vehicles for a given solution. Secondly, it calls a method named VND to minimize the distance for each vehicle. We should notice that the total profit is always the same. For LS2, it tries to insert unsatisfied demands to increase the total profit. The priority of inserting the demand is defined by the distance between the demand and the given solution. The implementations are presented from Algorithm2 to Algorithm3 (see in Appendix section). In the algorithm, the BestInsertion(s, d) method tries to insert a demand d into route s in a way that the distance is minimized. The Shortest(s) finds the route in which the distance is minimal. The Distance(d, d′) method calculates the distance between two demands by using the equation defined below:

\[
\text{Distance}(d, d') = \|P_d - P_{d'}\| + \|D_d - D_{d'}\| \tag{7}
\]

Where \(P_d\) and \(D_d\) are the related pickup and delivery points for demand d.

**V. EXPERIMENTS**

In this section, a self-comparison is first made to show the performance of each method. Then a comparison with genetic algorithm from the literature is made to show the efficiency of the proposed method. The experiments are based on the benchmark proposed in [10].

**A. Running environment and parameters**

The programming language used is Java, and the method is run on an Intel Core i7-4810MQ CPU, 2.80 GHz processor with 16 GB of memory. The value of each parameter is determined after several tests, and is shown in Table I.

<table>
<thead>
<tr>
<th>W</th>
<th>N</th>
<th>Iteration</th>
<th>α</th>
<th>σsh</th>
<th>MAXNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>50</td>
<td>100</td>
<td>1</td>
<td>377</td>
<td>10</td>
</tr>
</tbody>
</table>

**TABLE I:** parameters

**B. Self-comparison**

Since the proposed method is a hybridization of PSO and local searches, a validation of each method is necessary. The tests are made on the benchmarks, and the result of instance SPDPTWD-R201 is shown in Fig.6. In the figure, pure PSO has evolved from the initial population. This can justify that PSO itself has the capacity of improving the results. However, the result is still a bit far from the real pareto front. After integrating local searches, the result obtained by the method called HPSO is closer to the real pareto front (see in section III for more information). This can validate the performance of the proposed local searches. This phenomenon can be observed for all the tested instances.

**C. Comparison with other method**

The second comparison is made between HPSO and HG which is proposed in [12]. The comparison is also made between HPSO and real pareto front/approached pareto front. The instances are divided into two groups. In the first group, all the real pareto fronts are obtained. In the second group,
the pareto fronts are approximated. Three criteria are taken into account to make comparison: 1) Gap; 2) Number of non-dominated solutions; 3) Solving time. The gap is calculated using the equation defined below:
\[
\text{Gap} = \frac{\sum_{i=1}^{Nb} \left[ \text{Distance}_{i,HPSO} - \text{Distance}_{i,HG} \right] 
\times 100}{Nb}
\]
Where \(Nb\) is a set of common solutions (same profit) found by HPSO and HG. Distance\(i\) is the distance of solution \(i\), where \(i \in Nb\).

The results of the first group are shown in Table II. In terms of gap, HPSO can perform better in two instances than HG, and is near the real pareto front. As to the number of solutions, HPSO can also find more solutions than HG. The solving time of HPSO is less than HG.

The results of the second group are shown in Table III. In terms of gap, HPSO performs better than HG in two instances. In eight instances, the gap is almost 0%. Compared to approached pareto front, five instances achieve the gap between 2% and 4.6%. In other instances, a gap around 1% is achieved except for the instance SPDP-TWPD-R1002 in which the gap is 18% compared to HG and 15% compared to approached pareto front. As to the number of solutions and solving time, HPSO is always better than HG.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Gap(%)</th>
<th>Number of solutions</th>
<th>Solving time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDP-TWPD-C201</td>
<td>2.11</td>
<td>13.98</td>
<td>12</td>
</tr>
<tr>
<td>SPDP-TWPD-C202-1</td>
<td>0.43</td>
<td>2.91</td>
<td>32</td>
</tr>
<tr>
<td>SPDP-TWPD-C202-2</td>
<td>0.07</td>
<td>0.85</td>
<td>16</td>
</tr>
<tr>
<td>SPDP-TWPD-R201</td>
<td>-1.27</td>
<td>1.15</td>
<td>26</td>
</tr>
<tr>
<td>SPDP-TWPD-R202</td>
<td>-1.27</td>
<td>1.15</td>
<td>26</td>
</tr>
<tr>
<td>SPDP-TWPD-RC201</td>
<td>1.64</td>
<td>2.66</td>
<td>32</td>
</tr>
<tr>
<td>SPDP-TWPD-RC202</td>
<td>-0.63</td>
<td>0.45</td>
<td>19</td>
</tr>
</tbody>
</table>

TABLE II: First group

<table>
<thead>
<tr>
<th>Instance</th>
<th>Gap(%)</th>
<th>Number of solutions</th>
<th>Solving time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDP-TWPD-C201</td>
<td>2.11</td>
<td>13.98</td>
<td>12</td>
</tr>
<tr>
<td>SPDP-TWPD-C202-1</td>
<td>0.43</td>
<td>2.91</td>
<td>32</td>
</tr>
<tr>
<td>SPDP-TWPD-C202-2</td>
<td>0.07</td>
<td>0.85</td>
<td>16</td>
</tr>
<tr>
<td>SPDP-TWPD-R201</td>
<td>-1.27</td>
<td>1.15</td>
<td>26</td>
</tr>
<tr>
<td>SPDP-TWPD-R202</td>
<td>-1.27</td>
<td>1.15</td>
<td>26</td>
</tr>
<tr>
<td>SPDP-TWPD-RC201</td>
<td>1.64</td>
<td>2.66</td>
<td>32</td>
</tr>
<tr>
<td>SPDP-TWPD-RC202</td>
<td>-0.63</td>
<td>0.45</td>
<td>19</td>
</tr>
</tbody>
</table>

TABLE III: Second group

The obtained gap for the instance SPDP-TWPD-R1002 is maybe due to the way of comparison which compares only the solutions of same profit. For this instance, the number of common solutions maybe very small, and the solutions obtained by HG are significantly better than HPSO. It will then give us the illusion that HG performs better than HPSO. However, HPSO may perform better than HG for the other solutions non-compared. In terms of number of solutions, HPSO always performs better than HG. It maybe due to the fact that HPSO stores a copy of the solutions before the apply of local searches, which allows HPSO to keep the solution of low profit and high profit at the same time. As to the solving time, HPSO is faster than HG thanks to the simplicity of encoding and decoding of PSO.

VI. CONCLUSION

In this paper, a problem called SPDP-TWPD has been studied. Since the optimal solutions of large instances are very hard to obtain, an approached method hybridizing PSO with local searches has been proposed. The idea is to take advantage of the two integrating methods: the capacity of global search of PSO, and the capacity of intensive search of local search. The experiment has been made on benchmarks from the literature. The results have shown that the proposed method can obtain high quality solutions in short time. Another remark from the conducted experiments is that PSO needs the help of local search to find better solution. In the future works and to improve the results, more effective local searches will be studied. To deal with much more problems in real life, other objectives or variants may also be considered.

VII. ACKNOWLEDGMENTS

This work is supported by the ANR (French National Research Agency) in the framework of the project TCDU (Collaborative Transportation in Urban Distribution). This project ANR-14-CE22-0017 is labelled by the Pôle Véhicule du Futur, and is jointly performed by four partners, the three french universities of technology (UTT, UTBM, UTC) and the society Share And Move Solutions.

REFERENCES


VIII. APPENDIX

Algorithm 4: VND

Input : Solution S, Unsatisfied demands D;
Output : Solution S, Unsatisfied demands D;
for each route r in S do
    distance = Distance(r);
    Stock demands of r into list l;
    count = 0;
    while count < l.length do
        d = list.get(count);
        Stock demands which have the same profit as d from D into list l';
        if l'.isEmpty() then
            count = count + 1;
            continue;
        end
        flag = false;
        for each demand d' in l' do
            r' = r;
            r'.remove(d');
            D.remove(d');
            D.add(d);
            l.set(count, d');
            if BestInsertion(r', d') then
                temp = Distance(r');
                if temp < distance then
                    r = r';
                    flag = true;
                    distance = temp;
                end
            else
                D.remove(d);
                D.add(d');
                l.set(count, d);
                r' = r;
            end
        end
        count = count + 1;
        if flag == true then
            break;
        end
    end
end
VND(); //Algorithm 4

Algorithm 3: LS2

Input : Solution S, Unsatisfied demands D;
Output : Solution S, Unsatisfied demands D;
Stock demands of S into list l;
for each demand d in D do
    distance = ∞;
    for each demand d' in l do
        temp = Distance(d, d');
        if temp < distance then
            distance = temp;
        end
    end
end
Sort D in ascending order in terms of distance;
for each demand d in D do
    if BestInsertion(S, d) == true then
        D.remove(d);
    end
end