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.

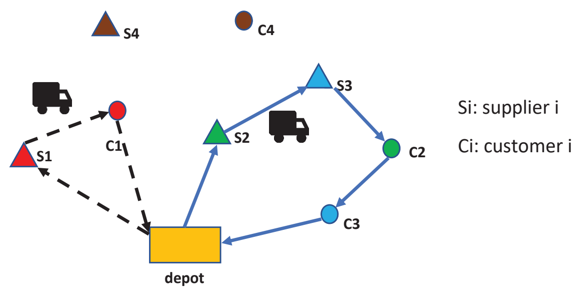

Fig. 1. Example of SPDPTWPD.

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 $profit_r$ can be gained if demand r with quantity q_r is satisfied. s_r is the loading/unloading time at the pickup/delivery point of demand r. The problem can be defined on a complete directed graph $G = (N, E)$, where $N = W \cup P \cup D$ is the set of all sites, W is the depot. E is the set of arcs where $E = \{(n_i, n_j) | n_i, n_j \in N, i \neq j\}$. Each site $i \in N$ is associated with a time window $[e_i, l_i]$, where e_i is the earliest arrival time, and l_i is the latest arrival time. K is the set of vehicles. Each vehicle $k \in K$ has a capacity Q_k. When a vehicle is making its tour, the following constraints need to be respected: 1) A pickup point should always be visited before the related delivery point; 2) The time window of each site should be respected; 3) The capacity of a vehicle should not be violated; 4) A vehicle should depart and return back to the same depot. A mixed integer linear program has been introduced in [9]. Two objectives (maximizing the profit/minimizing the distance) have been considered in [10], in which related instances were proposed. Since the authors have just generated only one non dominated solution (the one with maximum profit), we are interested in generating the complete pareto front for each instance.

A solution of the problem consists of a profit and a distance. By fixing the profit, a related distance can be obtained. The aim is to generate the complete Pareto front from all the combinations of profit. The model is run on a powerful server with 128 processors and 264Go memories. For all the instances with 20 sites, the complete pareto fronts are obtained. However, For instances with 50, 100 sites, it is very difficult to get optimal solution. As consequence, the solving time for each profit has been limited to get approximated pareto fronts. The results can be found on the site: https://www.dropbox.com/s/ri1v9wffjrdahtr/pareto%20optimal.xlsx?dl=0.

IV. Hybrid Multi-Objective Particle Swarm Optimization

In this section, the principles of Multi-Objective PSO (MOPSO) is firstly introduced. Then, a problem related encoding strategy is presented. At last, local searches are detailed.

A. Multi-Objective Particle Swarm Optimization

Inspiring from the behavior of flock of birds, PSO has been used to solve many optimization problems since it was proposed. It can be described as a group of particles in which each represents a solution. Each particle has a memory which can record the best solution ever found by itself (Pbest). Among these best solutions, a leader is selected to present the best solution ever found by the group (Gbest). The evolution of each particle is effected 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 solve the studied problem. Before the adaptation, a few modifications are made. First, let us introduce two equations:

$$\begin{aligned} VEL_i = W * VEL_i + R_1 * (REP\_PBEST_i[h] - POP_i) \\ + R_2 * (REP\_GBEST_i[h] - POP_i) \quad (1) \\ POP_i = POP_i + VEL_i \quad (2) \end{aligned}$$

For particle i, VEL_i represents its velocity, and POP_i represents its position (solution). REP_PBEST_i is the repository recording all the non dominated solutions ever found by itself. REP_GBEST_i is the repository recording all the non dominated solutions found by the group. h is the index of selected solution. W is the inertia weight. R_1, R_2 are random numbers in the range $[0, \ldots, 1]$.

Repository can be seen as an external file used to record non dominated solutions. Three related operations need to be defined: update, capacity, selection. If a new candidate is dominated by any solution in the repository, it will not be considered. If it dominates any solution, it is recorded, and all the 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. In FIFO, the selection procedure is based on the sequence by which each candidate enters the repository. However, this procedure would be disturbed when updating the repository. In this case, a corresponding strategy is illustrated in Fig.2. Iterator points the selected solution, and it moves from left to right. When it reaches the tail, it will restart from the head.

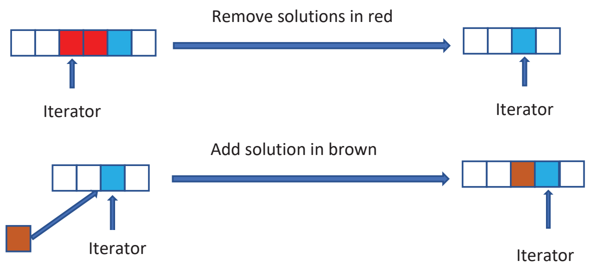

Fig. 2. FIFO strategy

The second strategy is based on the idea that solutions in a less crowded area have higher chance to be selected. For each solution i in the $population$ (repository), a probability can be calculated:

$$Prob_i = \frac{Fitness_i}{\sum_{j \in population} Fitness_j} \quad (3)$$

Where $Fitness_i$ is the fitness of each solution, it can be calculated:

$$Fitness_i = \frac{1}{\sum_{j \in population} \phi_{ij}} \quad (4)$$

Where ϕ_{ij} is the density which can be calculated:

$$\phi_{ij} = \begin{cases} 1 - \left(\frac{dist_{ij}}{\sigma_{sh}}\right)^{\alpha}, & dist_{ij} < \sigma_{sh} \\ 0, & Otherwise \end{cases} \quad (5)$$

Where α and σ_{sh} are the sharing parameters. $dist_{ij}$ is the Euclidean distance between solutions i and j, which can be calculated:

$$dist_{ij} = \sqrt{(s_i^1 - s_j^1)^2 + \ldots + (s_i^n - s_j^n)^2} \quad (6)$$

Where $(s_i^1, \ldots s_i^n)$ is solution vector i, with n corresponding to the number of objectives. As we can see from the equations defined above, the solution with higher fitness value has more chance to be selected.

Based on the preliminary experiments, FIFO is used for REP_PBEST_i, and 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.

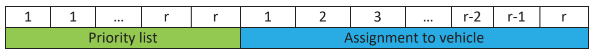

Fig. 3. Representation of solution

In the figure, a particle is divided into two parts. In the first part, each demand in $[1 \ldots r]$ is assigned with two dimensions. The value of each dimension is randomly generated in $[0 \ldots 10]$. 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.

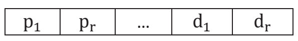

Fig. 4. Visiting sequence A

In the second part, each demand is assigned with one dimension. The value of each dimension is randomly generated in $[0, \ldots, NV[$, 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

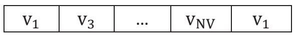

Fig. 5. Vehicle assignment

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 $D$;
  $D = \emptyset$;
  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
  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:

$$Distance(d, d^{'}) = \|P_d - P_{d'}\| + \|D_d - D_{d'}\| \qquad (7)$$

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.

W	N	Iteration	α	σ_{sh}	MAXNUM
0.8	50	100	1	377	10

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 SPDPTWPD-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.

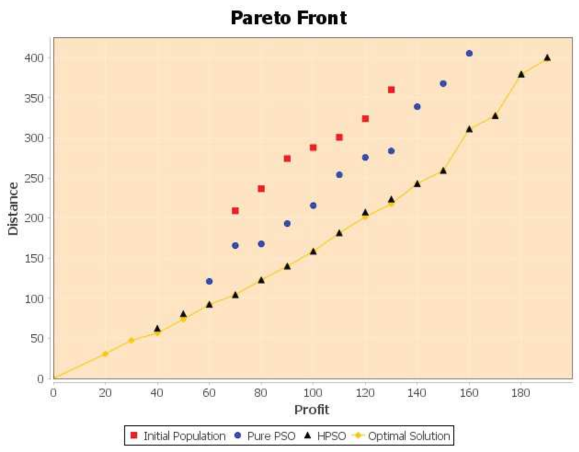

Fig. 6. Self-comparison.

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:

$$Gap = (\sum_{i=1}^{Nb} \frac{Distance^i_{HPSO} - Distance^i_{HG}}{Distance^i_{HG}} * 100)/Nb \quad (8)$$

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, HSPO can also find more solutions than HG. The solving time of HPSO is less than HG.

Instance	Gap(%)		Number of solutions			Solving time(s)	
	HPSO/HG	HPSO/Optimal	HPSO	HG	Optimal	HPSO	HG
SPDPTWPD-C201	2.13	1.58	12	10	18	3.06	17.00
SPDPTWPD-C202-1	0.43	0.91	26	7	29	2.27	13.12
SPDPTWPD-C202-2	0.43	2.91	32	8	43	3.94	15.47
SPDPTWPD-R201	0.07	0.85	16	9	19	1.17	12.45
SPDPTWPD-R202	-1.27	1.15	26	6	33	1.59	13.01
SPDPTWPD-RC201	1.64	2.66	16	9	17	1.33	12.39
SPDPTWPD-RC202	-0.63	0.45	19	6	24	1.54	12.82

TABLE II: First group

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 SPDPTWPD-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.

Instance	Gap(%)		Number of solutions			Solving time(s)	
	HPSO/HG	HPSO/Approached	HPSO	HG	Optimal	HPSO	HG
SPDPTWPD-C501	0.01	0.69	18	6	47	42.01	63.87
SPDPTWPD-C502-1	0.01	3.13	27	5	70	6.32	442.15
SPDPTWPD-C502-2	0.01	3.83	33	8	117	12.63	484.71
SPDPTWPD-R501	0.00	0.76	27	6	49	66.34	32.87
SPDPTWPD-R502	-2.98	3.40	78	6	120	20.00	22.13
SPDPTWPD-RC501	1.86	1.76	22	5	32	3.12	31.14
SPDPTWPD-RC502-4	0.00	0.38	46	5	71	1.00	27.48
SPDPTWPD-RC502-5	-1.13	1.11	55	6	92	1.14	29.01
SPDPTWPD-RC502-6	2.24	0.83	56	9	108	1.56	29.98
SPDPTWPD-C1001	0.01	0.79	17	6	64	50.02	310.79
SPDPTWPD-C1002	4.28	4.09	24	7	73	189.15	256.57
SPDPTWPD-R1001	4.47	4.62	26	5	44	20.01	152.95
SPDPTWPD-R1002	18.13	14.75	65	6	135	37.02	183.95
SPDPTWPD-RC1001	2.31	1.49	11	5	13	5.30	169.08
SPDPTWPD-RC1002-1	0.01	0.18	12	5	17	1.02	112.54
SPDPTWPD-RC1002-2	0.01	2.18	27	6	40	2.33	166.24

TABLE III: Second group

The obtained gap for the instance SPDPTWPD-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 SPDPTWPD 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

[1] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. (2016) The vehicle routing problem: State of the art classification and review.

[2] P. Toth and D. Vigo. (2002) The vehicle routing problem (society for industrial and applied mathematics, philadelphia).

[3] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems," *Journal für Betriebswirtschaft*, vol. 58, no. 1, pp. 21–51, Apr 2008. [Online]. Available: https://doi.org/10.1007/s11301-008-0033-7

[4] S. N. Parragh, K. F. Doerner, and R. F. Hartl. (2007) A survey on pickup and delivery problems.

[5] S. Ropke and J.-F. Cordeau. (2009) Branch and cut and price for the pickup and delivery problem with time windows.

[6] R. Baldacci, E. Bartolini, and A. Mingozzi. (2011) An exact algorithm for the pickup and delivery problem with time windows.

[7] H. Lim, A. Lim, and B. Rodrigues. (2002) Solving the pickup and delivery problem with time windows using issqueaky wheelli optimization with local search.

[8] H. Li and A. Lim. (2003) A metaheuristic for the pickup and delivery problem with time windows.

[9] Z. Al Chami, H. Manier, and M.-A. Manier, "New model for a variant of pick up and delivery problem," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 001 708–001 713.

[10] Z. Al Chami, H. Manier, and M.-A. Manier, "A lexicographic approach for the bi-objective selective pickup and delivery problem with time windows and paired demands," *Annals of Operations Research*, Apr 2017. [Online]. Available: https://doi.org/10.1007/s10479-017-2500-9

[11] Z. Al Chami, H. El Flity, H. Manier, and M.-A. Manier, "A new metaheuristic to solve a selective pickup and delivery problem," in *2018 4th International Conference on Logistics Operations Management (GOL)*, April 2018, pp. 1–5.

[12] Z. Al Chami, H. Manier, M.-A. Manier, and C. Fitouri, “A hybrid genetic algorithm to solve a multi-objective pickup and delivery problem,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14 656 – 14 661, 2017, 20th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317325326
[13] Z. Peng, H. Manier, and M.-A. Manier, “Particle swarm optimization for capacitated location-routing problem,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14 668 – 14 673, 2017, 20th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317334134
[14] V. Kachitvichyanukul, P. Sombuntham, and S. Kunnapapdeelert. (2015) Two solution representations for solving multi-depot vehicle routing problem with multiple pickup and delivery requests via pso.
[15] K. Govindan, A. Jafarian, R. Khodaverdi, and K. Devika. (2014) Two-echelon multiple-vehicle location–routing problem with time windows for optimization of sustainable supply chain network of perishable food.
[16] M. Reyes-Sierra, C. C. Coello *et al.*, “Multi-objective particle swarm optimizers: A survey of the state-of-the-art,” *International journal of computational intelligence research*, vol. 2, no. 3, pp. 287–308, 2006.
[17] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, “Handling multiple objectives with particle swarm optimization,” *IEEE Transactions on evolutionary computation*, vol. 8, no. 3, pp. 256–279, 2004.

VIII. Appendix

Algorithm 4: VND

```
Input : Solution $S$, Unsatisfied demands $D$;
Output : Solution $S$, Unsatisfied demands $D$;
for each route $r$ in $S$ do
    $distance = Distance(r)$;
    Stcok 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 $l'.isEmpty()$==true 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 $BestInsertion(r', d') == true$ 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
        end
        $count = count + 1$;
        if $flag == true$ then
            $count = 0$;
        end
    end
end
```

Algorithm 2: LS1

```
Input : Solution $S$, Unsatisfied demands $D$, Number of
  iterations $MAXNUM$;
Output : Solution $S$, Unsatisfied demands $D$;
$count = 0, S' = S, D' = D$;
while count<MAXNUM do
    $S'' = S$;
    $D'' = D$;
    Route $r = Shortest\_route(S)$;
    Stock demands of $r$ into list $l$;
    Delete $r$ from $S$;
    $flag = true$;
    for each demand $d$ in $l$ do
        $flag = BestInsertion(S, d)$;
        if flag==false then
            Stock demands which have the same profit as $d$
              from $D$ into list $l'$;
            for each demand $d'$ in $l'$ do
                $flag = BestInsertion(S, d')$;
                if flag==true then
                    $D.add(d)$;
                    $D.remove(d')$;
                    break;
                end
            end
            if flag==false then
                break;
            end
        end
    end
    if flag==false then
        $S = S''$;
        $D = D''$;
        break;
    end
    $count = count + 1$;
end
VND(); //Algorithm 4
if $Distance(S) > Distance(S')$ then
    $S = S'$;
    $D = D'$;
end
```

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
    $distance_d = \infty$;
    for each demand $d'$ in $l$ do
        $temp = Distance(d, d')$;
        if $temp < distance_d$ then
            $distance_d = temp$;
        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
```