Hybrid metaheuristic for the Pickup and Delivery Problem designed for passengers and goods transportation

Alexis Godart[1], Hervé Manier[1], Christelle Bloch[2], Marie-Ange Manier[1]

[1] *Univ. Bourgogne Franche-Comté, FEMTO-ST Institute/CNRS, Rue Thierry-Mieg (UTBM), 90010 Belfort Cedex, France (e-mail: {alexis.godart, herve.manier, marie-ange.manier}@utbm.fr,).*
[2] *Univ. Bourgogne Franche-Comté, FEMTO-ST Institute/CNRS, 1 Cours Leprince-Ringuet, 25200 Montbéliard, France (e-mail:christelle.bloch@univ-fcomte.fr)*

Abstract: This paper introduces a new formulation for a variant of static Pickup and Delivery Problem involving passengers and goods, time windows, multiple visits and transfer operations with or without storage. This model contributes to address mobility and logistics requirements for stakeholders with the rise of Smart Cities. Solutions of good quality are found thanks to a hybrid metaheuristic based on an evolutionary algorithm. Our results are compared on small instances for which optimal solutions have been found using MILP. We discuss perspectives and upcoming challenges such as the scalability of the model.

Keywords: Pickup and Delivery; Multimodal; Dial-A-Ride; Transfers; Hybrid metaheuristic; Evolutionary algorithm; Smart City

1. INTRODUCTION

This paper introduces a new variant of Pickup and Delivery Problem (PDP) subject to urban-related constraints. The approach presented is based on a hybrid evolutionary algorithm to provide solutions of good quality on some instances. That work is part of a french project called MISC (Mobility In Smart Cities). The main purpose of this project is to create a platform providing real-time mobility services for both people and goods transportation. It focuses on finding a compromise between keeping cost control and maximizing quality of service for all mobility stakeholders (carriers, users, customers) but also local authorities. Cohabitation opportunities between passengers and goods in a same network are an upcoming trend. They are mainly based on similarities of constraints, and reviewed in Trentini and Mahléné (2010) and Trentini and Mahléné (2012). In addition, some authors explore these features, such as Masson et al. (2017). The urban transport problem is often referred to as "last mile logistics" and represents a major interest in transportation research. Ranieri et al. (2018) give a review based on recent scientific contributions and focus on externalities cost reduction. Several variants of PDP have been explored, including characteristics like soft/hard time windows constraints, multiple echelons, multiple periods or multiple trips per vehicle. This abundant literature offers a wide variety of models that can handle most of modern and specific transportation problems. However, urban transport is unique in its particularities. Vehicles travel short distances and perform numerous and frequent operations, all into a dense and overcrowded infrastructure. Multi-echelons PDP also focuses on the urban context, but it considers restrictions regarding the access of vehicles. This assumption is often interesting in the case of restricted-area such as city center, but sometimes it is also used to simplify the problem.
In this paper, vehicles are assumed allowed to travel everywhere, leaving the possibility of transfer between two or more modes. Each vehicle can visit a site multiple times during its trip. The aim of this work is to measure effectiveness with such an approach for an integrated transportation problem involving a lot of interdependancies. This problem is known for its $\mathcal{NP}$-hardness which requires efficient strategies to tackle large instances.

2. LITERATURE REVIEW

This section describes some papers that are relevant to our problem, the static PDP with multiple visits and transfers. Our paper research extends to Dial-A-Ride Problem (DARP) because of its similarities and inspiration found for this problem. Variants exploration is limited to those containing time windows and/or transfers with and without synchronization.

2.1 Pickup and Delivery Problem

The reader can refer to both Parragh et al. (2008) and Berbeglia et al. (2007) for a general framework on models and algorithms used for PDP, and to Ropke et al. (2007) for models of PDP with time windows (PDPTW). One formulation of PDP can be as follows: a fleet of vehicles must satisfy a set of transportation demands between sites, subject to some transportation-dependent constraints.

Ghilas et al. (2016) addresses a variant of PDPTW which includes scheduled public transportation lines. Al Chami et al. (2018) presents a new variant called the multi-period Pickup and Delivery Problem with Time Windows and Paired Demands (Mu-PDPTWPD) which suits well in the context of urban distribution. Considering multiple periods can lead to great savings because of improved tactical scheduling. Masson et al. (2017) introduces a two tiered transportation problem. The problem formulation implicitly describes a system that mixes passengers and goods. First tier allows goods to be transported in city buses from a consolidation and distribution center to a set of bus stops originally used for passengers. In the second tier, final customers are distributed by a fleet of city freighters after transferring goods. A case study is proposed onto a medium-sized city.

Ghilas et al. (2016) efficiently adapt an Adaptive Large Neighborhood Search (ALNS) to find solutions of good quality for some generated instances, dealing with instances containing up to 100 freight requests in one hand. Results obtained on a real-life scheduled line system show the efficiency in terms of quality of solutions and computational time. Al Chami et al. (2018) use an advanced Greedy Randomized Adaptive Search Procedure (GRASP) combined with an Hybrid Genetic Algorithm (HGA) to solve instances containing up to 10 periods. Masson et al. (2017) also use an ALNS algorithm to tackle 15 generated data sets and provide a comparison of transport systems emphasize risks about bad synchronization.

One or more objective in PDP are often expressed with operational costs, distance, profits, waiting time and so on. In Al Chami et al. (2018) the problem variant is solved by minimizing simultaneously the total traveled distance by vehicles, but also delays by introducing penalties. Objective in Ghilas et al. (2016) is the total travel cost based on vehicles dedicated for pickup and deliveries and vehicles dedicated for transferred requests.

2.2 Dial-A-Ride Problem

DARP is fundamentally similar to PDP in its modeling and the reader can refer to Cordeau and Laporte (2007) for a general framework on DARP. It deals with on demand transport (mostly expressed by passengers). Objective functions are more oriented towards quality of service, mostly seen by minimizing delays, waiting time or ride time. A general framework for DARP with transfers (DARPT) is addressed by Masson et al. (2014). The problem is solved with an ALNS using insertion techniques. Experiments on real-life instances show savings due to transfers up to 8%. Guerriero et al. (2014) solve a DARP depicting a public transportation system where each customer specifies his pickup point and drop-off point, and a requested time window. Their work contributes to find both the maximum total ride time and the total waiting time. The problem is solved with some metaheuristics. Parragh and Schmid (2013) introduce a DARP in which total routing costs are minimized while routes must respect time window, maximum user ride time, maximum route duration, and vehicle capacity restrictions. Problem is tackled using a hybrid column generation and a large neighborhood search algorithm.

These papers emphasize very interesting features and upcoming trends. Our goal is to consider several ones simultaneously and to find good approaches to solve even more complex instances. Some interesting properties are described in the following section.

3. PROBLEM FORMULATION

This section describes the PDP variant tackled in this paper. This variant is solved to optimality in Godart et al. (2019) with exact methods using Mixed Integer Linear Programming (MILP) and implemented on IBM CPlex$^{\text{TM}}$ solver. The model created is greatly inspired by Cortés et al. (2009) work and Masson et al. (2014) work.

3.1 Problem description and modelling

Transportation demands $r \in R$ are all known in advance (static demands). All of them must be satisfied within their respective time windows $[p_r^+; p_r^-]$ and they cannot be split up. Each demand r corresponds to q_r entities moving from the origin node S_r^+ to the destination node S_r^-. These entities can be either goods or passengers. In order to separate or mix goods and passengers in a mode, a set B_r contains all vehicles $v \in V$ able to handle r. As an example, a vehicle dedicated to the transportation of goods can appear only in $B_{r'}$ such as $r' \in R$ is a demand concerning goods. Service times are defined for both pickup δ_r^+ (loading operation) and drop-off δ_r^- (unloading operation) associated with demand $r \in R$.

Vehicles v are defined in a set V with heterogeneous capacities Q_v. Each vehicle is either dedicated to the passengers mobility or to the goods transportation, which means its capacity is dedicated to one type of entities. Vehicles have heterogeneous speeds $speed_v$ equal to their mean velocity. They also have an associated depot vw_v from which the vehicle v begins and ends its trip.

The set W contains vehicle depots w. Set Ω designates origin/destination sites. Any site can possibly be origin and destination as it can be involved in more than one demand. Set T defines transfer points t with a storage capacity sc_t. This capacity is only used to deal with goods transportation. For passengers, capacity is unlimited in public places. No storage capacity (i.e equals to 0) implies a transfer operation in which both vehicles are meeting during a same common time window.

Finally, pickup or delivery operation on a node i must be realized within its opening time window $[e_i; l_i]$. A cost d_{ij} is assigned to each pair of nodes (i, j), and represents the shortest distance between i and j in our context.

The 39 constraints in the original problem are described in Godart et al. (2019). They include strict time windows constraints for sites and demands, which can lead to infeasibility. Thus, we use a new version of the MILP that allows delays by relaxing the upper bound constraints on both nodes and demands time windows. These relaxations have been made to avoid bad or no results because of really low percentage of feasible solutions on some instances. This assumption intends to let the evolutionary algorithm explore the search space and potentially find some solutions that would be unreachable otherwise.

3.2 Objective function

Our study focuses on Quality of Service (QoS) and cost reduction. Then we have adopted a multi-objective approach by considering three objectives. The first one minimizes the overall distance traveled by all vehicles. The two other objectives correspond to the QoS. One minimizes the highest delay on demand satisfaction. If demand is satisfied on time, corresponding delay is null. Else, this delay is equal to the difference between the effective delivery date and the latest delivery date p_r^-. In the same way, the third objective minimizes the highest delay when visiting sites. These two criteria permit to guide the hybrid method towards solutions which strictly satisfy the time windows constraints (since the MILP model considers them as soft constraints). This multi-objective optimization transforms the hard time window constraints in soft ones, while indicating in the algorithm that the solutions with no delay are the most interesting ones.

4. METHODS

The original PDP deals with assignment of demands on vehicles and with vehicle routing. It includes time and capacity constraints, but also in our variant transfer operations between vehicles, both with and without synchronization. These two interdependent sub-problems are solved simultaneously in Godart et al. (2019). However, combinatorial complexity remains too important and prevents us from tackling large size instances.

Figure 1 shows the decomposition of the problem and presents the proposed hybrid meta-heuristic. The first step assigns the demands to vehicles into the evolutionary algorithm. Then second step assigns pickup, delivery and transfer operations to vehicles using a greedy algorithm. The last step creates feasible routes by solving a MILP. Some constraints are provided by the greedy algorithm assignment choices (step 2). The newly constrained linear program is used to solve the scheduling problem (i.e routing problem), while checking capacity and precedence constraints, and interdependencies between vehicles.

4.1 Global scheme of evolutionary algorithm

The evolutionary algorithm is based on the well-known NSGA-II by Deb et al. (2002). It has been adapted by many authors and its efficiency has been proven for several decision problems, especially vehicle routing based problems. This multi-objective algorithm ranks individuals in several Pareto fronts. It also uses crowding to both improve the quality and the diversity of the explored solutions. The evolutionary design used lies on a binary encoding (detailed in the following subsection). Classical operators are applied for the experimentation: a single-point crossover, with a probability of 90 %, and the BitFlip mutation, with a probability equal to one out of the size of the binary array. Finally, selection is managed by a binary tournament. Duplicate individuals are allowed.

4.2 Individuals encoding

In step 1, demand assignment is encoded with a binary matrix b, in which each element is defined as follows:

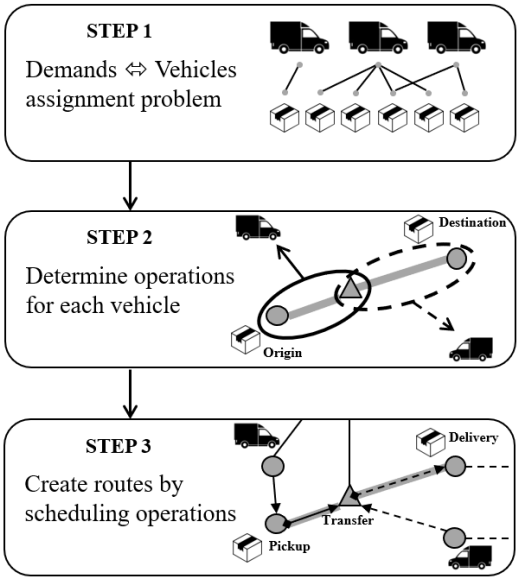

Fig. 1. Problem decomposition step-by-step

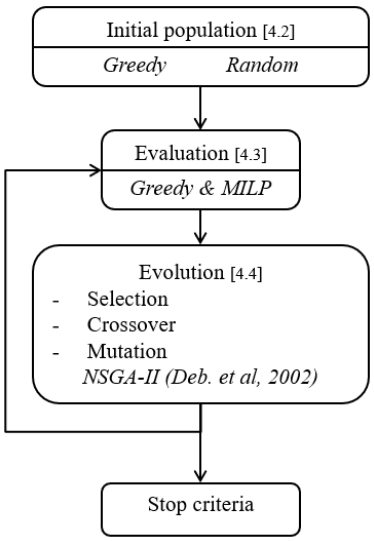

Fig. 2. Functional scheme of hybrid meta-heuristic

$$b_r^v = \begin{cases} \mathbf{1} \text{ if demand } r \in R \text{ is assigned to vehicle } v \in B_r, \\ \mathbf{0} \text{ else.} \end{cases}$$

Table 1 shows this matrix for a given example.

		Demands							
		1	2	3	4	5	6	7	8
Vehicles	1	1	0	0	0	1	1	0	0
	2	0	1	1	0	0	0	0	1
	3	0	0	1	0	0	1	0	1
	4	0	0	0	1	0	1	1	1

Table 1. Example of demand assignment with 4 vehicles and 8 demands

As all demands must be satisfied, each demand must have at least one vehicle $v \in B_r$ assigned (1). Several vehicles assigned to one demand implies transfer operations.

$$\sum_{v \in B_r} b_r^v \geq 1 \qquad \forall r \in R \tag{1}$$

4.3 Initial population

Let us note pop_i the set of individuals obtained after iteration i. Iteration 0 corresponds to the construction of the initial population pop_0. Each individual's genotype encodes the previously defined binary matrix b. Two algorithms are used to generate the initial population :

Random assignment: This algorithm randomly chooses one or several vehicles v in B_r for every demand $r \in R$.

Greedy algorithm: This algorithm (algorithm 1) assigns each demand to the best vehicle regarding three static indicators Lp_v^r, $Ltws_v^r$ and $Ltwr_v^r$. Those ones are a priori computed for each couple (r,v).

- Lp_v^r Physical proximity between vehicle depot w_v and origin/destination nodes S_r^+ and S_r^- (lower is better);
- $Ltws_v^r$ Time in common between the time window of vehicle depot and those of origin/destination sites (higher is better);
- $Ltwr_v^r$ Time in common between the time window of vehicle depot and request time window (higher is better).

Those indicators are weighted, and then used in a normalized score GS_v^r. Each request is then associated with the vehicle minimizing this score. Diversity in the initial population is increased by varying the weights from one individual to another. The generated solutions do not include transfer operations.

4.4 Evaluation function

Each individual, which encodes the assignment of demands on vehicles (step 1), is then evaluated using a process based on a greedy algorithm (step 2) and MILP (step 3) (see figure 1). This section explains the rules used to assign operations to each vehicle. For each vehicle $v \in V$, a set of assigned pickup operations op_v and a set of assigned delivery operations od_v are defined. Indeed, in case of transfers, each vehicle assigned to a demand must perform a part of the demand's route (figure 3). Therefore, the corresponding pick-up and drop-off operations must be assigned to the concerned vehicles.

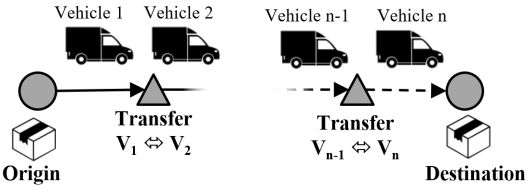

Fig. 3. Example of a demand assigned to n vehicles

At this stage, another greedy algorithm determines which part of the demand's route will be done by each assigned vehicle. Like in the previous greedy algorithm, two static indicators are calculated :common distance and time. A first set GSP_v^r stores normalized indicators between vw_v and pickup site S_r^+. The second set GSD_v^r gives normalized indicators between vw_v and delivery site S_r^- (algorithm 2). Set $affect_r$ contains all vehicles to which demand r is assigned.

Algorithm 1 Greedy algorithm to create individual o

Create random individual *in: p, tw out: b*
// calculate priority ratios
set $pf = p \ / \ (p + tw)$
set $twf = tw \ / \ (p + tw)$
// normalize time windows parameter
set $twratio$ = `max`(d) / (`max`(l) -`min`(e))
// calculate score for each pair 'request r assigned to vehicle v'
for $r = 1$ to `length`(R) **do**
  for $v = 1$ to `length`(V) **do**
    set $Lp_v^r = d_{ki} + d_{kj}$ $k = w_v$ $i = S_r^+$ $j = S_r^-$
    set $Ltws_v^r$ = `min`$(l_i; l_k)$ - `max`$(e_i; e_k)$
      + `min`$(l_j; l_k)$ - `max`$(e_j; e_k)$
      $k = w_v$ $i = S_r^+$ $j = S_r^-$
    set $Ltwr_v^r$ = `min`$(p_r^-; l_k)$ - `max`$(p_r^+; e_k)$ $k = w_v$
    if $v \notin B_r$ **then**
      set $GS_v^r = +\infty$
    else
      set $GS_v^r = Lp_v^r * pf$ - $(Ltws_v^r + Ltwr_v^r) * twratio * twf$
    end if
  end for
end for
// generate individual's array based on GS
for $r = 1$ to `length`(R) **do**
  set $best = +\infty$ `// Best` GS `value`
  set $vbest = 0$ `// Index of vehicle with best` GS `value`
  for $v = 1$ to `length`(V) **do**
    if $GS_v^r < best$ **then**
      set $best = GS_v^r$
      set $vbest = v$
    end if
  end for
  set $b_{vbest}^r = 1$
end for

The final order of tasks to do per demand is obtained in a set $vlist_r$. The first vehicle on the list picks up the request r while the second one deposits it. The transfer points are determined according to the physical proximity and the common opening time between the two vehicles and each transfer point. The method used here is based on previous algorithms with a normalized score on the two indicators. Each transfer point can welcome a demand once at most.

Every operation assignment is then submitted to MILP using constraints on decision variables. Fixing these decision variables helps to drastically reduce the research space and thus to accelerate the evaluation process. MILP execution run time is limited to one minute. If no solution has been found after that amount of time, high values are returned for all the three objectives.

5. RESULTS

First we detail the experimental environment. Algorithms in step 1 and step 2 and initial population strategies are written in Python. Route scheduling is ensured with MILP

implemented on IBM CPLEX®. Finally, the evolutionary algorithm NSGAII is written in java. Results are obtained on a Intel® Xeon® E7-4850 running 16 cores and 256 gigabytes of RAM.
Our hybrid method was tested on small generated instances, for which optimal solution has already been found using the MILP exact method. These instances and the corresponding solutions are summarized in table 2.

	F00	F01	F02	F03	F04
Vehicles	2	2	2	3	4
Demands	3	4	6	4	8
Best solution	3'532	3'437	3'828	2'979	6'478
% Gap	-	-	-	-	13.85
CPU (s)	1.7	0.8	3'118	132	1'209'600

Table 2. Best solutions obtained with exact methods (MILP, Godart et al. (2019))

The *best solution* line gives the distance travelled by vehicles with no delay. Solutions with no gap (F00-F03) are optimally solved while solution for F04 is an upper bound. For small instances, the size of population was set to 10 individuals, and only 50 individuals were examined in each run (iterations). Building an initial population which size would be upper or equal to the size of the search space would not be adequate to assess the quality of the evolution. Table 3 gives the parameter values used.

Algorithm 2 assigning operations to vehicles

Assign operations to vehicles *out: op, od*
// calculate priority ratios
// normalize time windows parameter
set $twratio$ = `max`(d) / (`max`(l) -`min`(e))
// calculate score for each pair 'request r assigned to vehicle v'
for $r = 1$ to `length`(R) **do**
  for $v = 1$ to `length`($affect_r$) **do**
    set $Lpp_v^r = d_{ki}$ $k = w_v$ $i = S_r^+$
    set $Lpd_v^r = d_{kj}$ $k = w_v$ $j = S_r^-$
    set $Ltwsp_v^r$ = `min`($l_i; l_k$) - `max`($e_i; e_k$) $k = w_v$ $i = S_r^+$
    set $Ltwsd_v^r$ = `min`($l_j; l_k$) - `max`($e_j; e_k$) $k = w_v$ $j = S_r^-$
    set $GSP_v^r = Lpp_v^r$ - $Ltwsp_v^r$ * $twratio$
    set $GSD_v^r = Lpd_v^r$ - $Ltwsd_v^r$ * $twratio$
  end for
  sort GSP and GSD `ascending`
  // Choose task per vehicle based on GSP and GSD
  set $iter = 1$
  // create task order of vehicles for demand r
  set $vp_r = \emptyset$
  set $vd_r = \emptyset$
  while `length`(GSP) > 0 **and** `length`(vp_r) <= `length`(T) **do**
    if GSP_{iter}^r <= GSD_{iter}^r **then**
      add `vehicle` $iter$ `at the end of` vp_r
    else
      add `vehicle` $iter$ `at the beginning of` vd_r
    end if
    remove `element` $iter$ `in` GSP
    remove `element` $iter$ `in` GSD
  end while
  set $vlist_r = vp_r \cup vd_r$
end for

	F00	F01	F02	F03	F04
Number of runs	10	10	5	5	5
Size of initial population	10	10	10	10	20
Number of iterations	50	50	50	50	300

Table 3. Parameters of the experimentation for "n" and "n+g" strategies

In order to measure the evolutionary algorithm effectiveness, we propose two experimentation plans. The difference lies in how initial population is built. The first one denoted "n" generates individuals using only random assignment strategy, while the second one named "n+g" also creates random individuals but inserts one with the greedy strategy. The idea is to give the evolutionary algorithm a good initial genetic patrimony, provided that the greedy algorithm leads to interesting assignment solutions. In table 4, the objective values only consider the set gathering each best solution without delay obtained per run.

"n"	Objective value		Gap BKS (%)		CPU (s)	
strategy	Best	Aver.	Best	Aver.	Best	Aver.
F00	3'532	3'648	0	3.3	25	26
F01	3'437	3'590	0	4.5	23	26
F02	*no sol.*	-	-	-	712	1'136
F03	3'190	3'452	7.1	15.9	31	34
F04	7'406	7'406	14.3	14.3	9'022	14'590
"n+g"	Objective value		Gap BKS (%)		CPU (s)	
strategy	Best	Aver.	Best	Aver.	Best	Aver.
F00	3'532	3'643	0	3.1	23	25
F01	3'437	3'437	0	0	22	24
F02	3'873	3'873	1.2	1.2	397	982
F03	2'979	2'979	0	0	29	32
F04	*no sol.*	-	-	-	13'461	23'725

Table 4. Distance and CPU time obtained for 5 instances with hybrid metaheuristic

For instance F02 with strategy "n" and instance F04 with strategy "n+g", no solution without delay has been found. Figure 4 shows solutions in the bi-objective context. Some solutions are close to or sometimes better than the BKS in term of distance (below 6200), but with a delay counterpart. When comparing the two strategies, "n+g" strategy seems to lead to solutions always close enough to the optimal one (1.2%). This can be explained by the greedy individual inserted into the initial population. In most of cases optimal solutions are found. Other solutions without delay are still close to the BKS with a mean gap reaching 4.5% at most. However, for some instances like F04, we find solutions at best without delay but with a greater distance.
We notice that the bigger the instance are, the higher the computational times are. This is due to the routing problem complexity which directly depends on the number of demands and vehicles. Nevertheless, our method enables us to obtain good solutions from 3 (F02) to 63 (F04) times faster than with the exact method.
The use of exact methods for the routing problem (MILP)

prevents the solving of large-scale instances as the combinatorial complexity grows exponentially with the number of demands and vehicles. Moreover, the exact optimization engine is invoked as many times as the vehicle routing problem must be solved, i.e. for each iteration, while large instances would require more iterations so that the solution space can be better explored.
Finally our work enabled us to test and validate our proposed method, in particular the partitioning of the problem into three sequential sub-problems, each one solved with approached (steps 1 and 2 in figure 1) or exact methods (step 3). Indeed the conducted tests show that despite this partition, we reach or get near of optimal solutions.

6. CONCLUSION

A hybrid metaheuristic combining a multi-objective evolutionary algorithm, greedy heuristics, and a MILP, has been developped to solve a Pickup and Delivery Problem in which the transfer of demands is allowed with or without synchronization. Our method decomposes the problem by delegating to the evolutionary algorithm the control of the exploration of solutions with transfers and their potential benefits. However, the evaluation of solutions with an exact method, like MILP, offers few perspectives for scale-up instances as the scheduling problem remains NP-hard. Main interests in interfacing exact methods such as MILP into a metaheuristic are to characterize the problem and explore the search space more efficiently, in the perspective of applying approximated methods. Relaxation of hard time windows and assignment heuristics let the evolutionary algorithm explore efficiently the space to find some of the rare feasible solutions. Nonetheless the choices made by the heuristics which assign the operations can be improved. The existing algorithm will allow to compare other original indicators, especially dynamic ones. This could bring a better assessment of each genotype. Then, the next step is to implement a faster evaluation function, based on these indicators embedded in heuristics such as insertion or neighborhood search. Also, solutions found in this paper can be used for benchmark purposes to compare future methods on this variant of PDP.

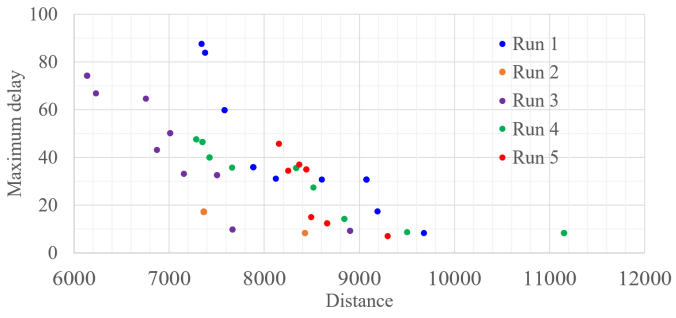

Fig. 4. Non-dominated solutions on 5 runs for instance F04 solved with "n+g" strategy.

ACKNOWLEDGEMENTS

This work is part of a Mobilitech project funded by the Bourgogne Franche-Comté region and labelled by Pôle Véhicule du Futur.

REFERENCES

Al Chami, Z., Manier, H., Manier, M.A., and Chebib, E. (2018). An advanced GRASP-HGA combination to solve a multi-period Pickup and Delivery Problem. *Expert Systems with Applications*.

Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1), 45–47.

Cordeau, J.F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1), 29–46.

Cortés, C.E., Matamala, M., and Contardo, C. (2009). The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197.

Ghilas, V., Demir, E., and Van Woensel, T. (2016). An adaptive large neighborhood search heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines. *Computers and Operations Research*.

Godart, A., Manier, H., Bloch, C., and Manier, M.A. (2019). MILP for a Variant of Pickup & Delivery Problem for both Passengers and Goods Transportation. In *Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018*, 2692–2698. IEEE.

Guerriero, F., Pezzella, F., Pisacane, O., and Trollini, L. (2014). Multi-objective optimization in dial-a-ride public transportation. *Transportation Research Procedia*, 3, 299–308.

Masson, R., Lehuédé, F., and Péton, O. (2014). The dial-a-ride problem with transfers. *Computers and Operations Research*, 41(1), 12–23.

Masson, R., Trentini, A., Lehuédé, F., Malhéné, N., Péton, O., and Tlahig, H. (2017). Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics*, 6(1), 81–109.

Parragh, S.N., Doerner, K.F., and Hartl, R.F. (2008). A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(2), 81–117.

Parragh, S.N. and Schmid, V. (2013). Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers and Operations Research*, 40(1), 490–497.

Ranieri, L., Digiesi, S., Silvestri, B., and Roccotelli, M. (2018). A review of last mile logistics innovations in an externalities cost reduction vision. *Sustainability (Switzerland)*, 10(3), 782.

Ropke, S., Cordeau, J.F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*.

Trentini, A. and Mahléné, N. (2010). Toward a shared urban transport system ensuring passengers & goods cohabitation. *Tema. Journal of Land Use, Mobility, and Environment*, 3(2), 37–44.

Trentini, A. and Mahléné, N. (2012). Flow Management of Passengers and Goods Coexisting in the Urban Environment: Conceptual and Operational Points of View. *Procedia - Social and Behavioral Sciences*, 39, 807–817.