A greedy based algorithm for a bi-objective Pickup and Delivery Problem with Transfers

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

***Abstract*— Optimization of urban transport is an ever-growing research area, especially with the massive success of e-commerce, the substantial demographic increase in urban areas, but also exploring interesting trends such as flexible multimodal itineraries. This work presents an innovative multi-stage approach where optimization methods such as greedy algorithms are adapted to solve a bi-objective vehicle routing problem with pickup and deliveries in urban areas with transfers. We highlight the performance of our approach on several instances with various sizes and characteristics.**

I. INTRODUCTION

With the development of e-commerce in particular, the issue of urban distribution is becoming increasingly important. How to deliver many parcels throughout a city, at different times and in a sustainable context? As the economic aspect is no longer the predominant criterion, the impact on the environment has become crucial with the development of more ecological transport solutions. The adopted solution aims to optimize the use of the transport resources, by adapting the capacity of the these ones to the volume of goods to be delivered or by adapting the type of energy used. Thus, the pooling of transport resources by grouping several requests and the search for the best routes are an integral part of the solution to the last mile delivery problem. In addition, to allow access to restricted traffic areas or to wait for staggered delivery times, it may be worthwhile to transship goods between two modes of transport, after a possible storage period at transfer points. In the scientific literature, this problem is similar to a Pickup and Delivery Problem with paired demands, time windows and transfers.

II. LITERATURE REVIEW

Pickup and Delivery Problems with Transfers take their roots on Vehicle Routing Problems family, known to be $\mathcal{NP}$-hard. Since its first introduction, Pickup and Delivery Problems with Transfers (PDPT) earned a lot of interest in research. In this section we describe some works relevant to our problem that can be found in literature. Masson, Lehuédé and Péton [6] proposed an Adaptive Large Neighborhood Search method to solve 780 instances containing from 50 to 100 requests each. In these instances, transfer points are known in advance. Karaoglan et al. [5] studied a location-routing problem with simultaneous pickup and delivery. Two mixed integer linear programming formulations are proposed to tackle small instances including an efficiency comparison, and they developed a heuristic algorithm based on simulated annealing to solve 360 instances of bigger scale. Mitrović and Laporte [9] used a transfer insertion heuristic to solve a PDPT in order to explore the benefit of transshipment. On demand transportation problems are defined as Dial-A-Ride Problems (DARP). They also arise in urban context, especially for transportation systems dedicated to elderly and disabled people. The reader can refer to Masson, Lehuédé and Péton [7] for a general framework on DARP. Deleplanque and Quillot [2] allow the insertion of dynamic transfer points and studied an algorithm based on insertion techniques and constraints propagation.

The few available literature dealing with such routing problems with transfers are mainly adressed using meta-heuristic methods and for mono-objective instances. We have also previously developped a hybrid genetic algorithm [12] but it is not performing fast enough to find feasible solutions with large instances. In this paper, we deal with a bi-objective problem for which we propose an algorithm based on several greedy algorithms involved at various decision levels.

III. PROBLEM DESCRIPTION

A. Notations

Let us consider a set of vehicles $v \in V$ with heterogeneous capacities $Q_v \in Q$ and heterogeneous speeds $s_v \in S$. Each vehicle v is associated to a vehicle depot $vw_v \in W$. The urban network under consideration is modelled with a graph $G = (N, E)$. The set of vehicle depots $w \in W$ represents all locations where associated vehicles start and end their route. Sites $z \in \Omega$ are used to express the origin and destination of requests, which are described later. Thus the set of nodes N is defined such that $N = W \cup \Omega$. The graph G is complete. Edges $\{i, j\} \in E$ link any node $i \in N$ with node $j \in N$. Its respective distance is denoted d_{ij}. A time window $[e_i; l_i]$ is assigned to each node $i \in N$. When expressed on a vehicle depot $w \in W$, the time window $[e_w; l_w]$ corresponds to the earliest date of vehicle departure e_w and to the latest date of vehicle return l_w. This statement is true for any vehicle $v \in V$ associated to this depot (i.e $vw_v = w$). When expressed for a site $z \in \Omega$, the time window $[e_z; l_z]$ corresponds to the opening time during which operations are allowed.

Finally, we define a set of transportation demands $r \in R$. A quantity q_r is associated to each demand r and has to be

*Mobilitech project funded by the Franche-Comté region (CPER 2015-2020) and labeled by Pôle Véhicule du Futur

[1]Univ. Bourgogne Franche-Comté, FEMTO-ST Institute/CNRS, Rue Thierry-Mieg (UTBM), 90010 Belfort Cedex, France (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 christelle.bloch@univ-fcomte.fr

moved from the origin node $S_r^+ \in \Omega$ to the destination node $S_r^- \in \Omega$. A set B_r lists vehicles that can handle r. Moreover, handling of demand $r \in R$ is allowed only within its time window $[er_r; lr_r]$. er_r is the earliest pickup date while lr_r is the latest delivery date. A service time for pickup δ_r^+ and for delivery δ_r^- is defined for any demand $r \in R$. Finally, a set T contains transfer points t where goods may change vehicle. Each transfer point t has a storage capacity sc_t.

B. Problem formulation

The problem to solve is non-selective as all demands $r \in R$ must be satisfied. The capacity of the vehicles must be respected: the load of a vehicle $v \in V$ cannot exceed its capacity Q_v at any time. We allow latest date violation on all time windows. However, vehicles are not allowed: (1) to leave their associated depot, (2) to visit sites or (3) to handle demand, before the earliest date of respective time windows. Thus time windows constraints can be defined as *[hard;soft]*. The number of vehicles is limited. Each vehicle at most performs one trip, so its use is not mandatory.

Our approach is to solve sequentially (1) the vehicle assignment problem then (2) the vehicle routing problem. These two problems are hierarchically interdependent. Changing solution for the assignment problem leads to a whole new vehicle routing problem to solve. Our first contribution is to provide a better understanding of the instance under consideration, based on the observation of static data. One thought would be to consider the integrated problem as a Location-Routing Problem (LRP). However the proposed approach does not fit exactly this category of problems. According to Nagy and Salhi in their review [10] p.650: [...] *We do not classify as belonging to the LRP an approach that deals with both location and routing aspects of a problem but does not address their inter-relation*. Because each vehicle is assigned to a vehicle depot, it seems interesting to explore strategies used for the facility location problem (FLP) which is part of LRP. Among these strategies, one does often appear in literature: clustering method. Mehrjerdi and Nadizadeh [8] used a greedy clustering method to solve simultaneously the FLP and VRP. Gao et al [3] have implemented a K-means clustering algorithm to solve the FLP.

C. Objective functions

We define two criteria to optimize: minimizing the total distance travelled by vehicles, and minimizing the highest delay on demands, on sites and on route endings (maximal tardiness). Solutions to the problem must answer the following question: what are the best trade-offs between reducing distance and introducing delays? To our knowledge, there is no way to verify beforehand if a solution with no delay (i.e feasible solution in the context of the *[hard;hard]* TW problem) really exists and so if it can be found for a given instance, because there is a fixed number of vehicles.

IV. A THREE STAGES ALGORITHM

Our approach is composed of three stages, each one addressing sequentially one or several sub-problems of the studied problem (Fig.1). The first stage uses three heuristic algorithms to solve the assignment problem. It generates an initial population of solutions. The second stage operates a fast and quite efficient routing algorithm. It is based on an iterative local search heuristic, exploring neighborhood on weights of the developed greedy algorithm. Then another greedy based algorithm builds routes for each vehicle, inspired by the first one in its mechanism. Vehicles do not transfer demands at this stage yet. In the third stage, we apply an improvement heuristic based on transfer insertion.

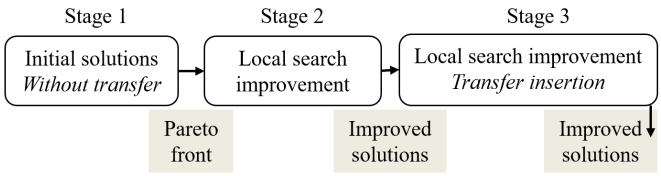

Fig. 1. Overview of the proposed approach

A. Stage 1

1) Assignment problem: Stage 1 first combines three heuristic procedures to generate an initial population. Those three heuristics work in a complementary way to cover largely the search space. Given a number of demands $|R|$ and a number of vehicles $|V|$, the assignment problem can be formulated as follows: which vehicle $v \in V$ is best suited to transport $r \in R$? With regard to the 2 objectives defined above, it does not seem possible to prove with certainty that one solution is better than another one. Indeed, a solution to this problem gives the assignment of vehicles to requests, but does not give a measure of the distance or delay potentially reachable by a routing algorithm.

Greedy1 algorithm

We have first elaborated greedy algorithm called Greedy1. Such an algorithm may be found in the literature, where clustering strategies are based on geographical position of sites and demands. However, time constraints bring a new dimension which is considered by our heuristics. We aim at providing not only good solutions but a quite wide Pareto front. One major advantage lies in its $O(n^2)$ complexity where $n = |R|$, which remains reasonable for this step in terms of computational time. Greedy1 assigns each demand to the best vehicle by concatenating three static indicators PS_v^r, TSS_v^r and TSR_v^r defined later. Those ones are a priori computed for each pair (r,v) (demand $r \in R$ and vehicle $v \in B_r$). Three weights wps, $wtss$ and $wtrs$ are submitted to the algorithm, associated with these static indicators. Varying these weights from one run to another intends to generate solutions which minimize either the distance, the maximal delay on demands or the maximal delay on sites.

PS_v^r = $d_{ij} + d_{ik}$, such as $i = vw_v$, $j = S_r^+$, $k = S_r^-$
Physical proximity indicator (lower is better)

TSS_v^r = min(l_i; l_j)-max(e_i; e_j)+min(l_i; l_k)-max(e_i; e_k)
such as $i = vw_v$, $j = S_r^+$, $k = S_r^-$
Common time window between the vehicle depot and the origin/destination sites (higher is better)

TRS_v^r = min(l_i; lr_r)-max(e_i; er_r)
such as $i = vw_v$, $j = S_r^+$, $k = S_r^-$
Common time window between the vehicle depot and the request (higher is better)

$maxd$ = max(d)
Normalization criterion for distance unit, equal to the highest distance among edges $\{i, j\} \in E$

$maxt$ = max(l)-min(e)
Normalization criterion for time unit, equal to the horizon of time of the instance

GS_v^r = $PS_v^r \cdot wps/maxd - (TSS_v^r \cdot wtss + TRS_v^r \cdot wtrs)$ $/maxt$
Concatenated score for pair (demand r, vehicle v)

Each request is then associated with the vehicle minimizing the concatenated score GS_v^r. In case of equality, one of the vehicles with the best score is randomly assigned.
Let us note that the indicators are based on static information. Thus spatio-temporal distances are calculated with a static reference point: the vehicle depot. However the vehicle being a dynamic element, it is reasonable to assume that indicators are not precise enough. For example, if two or more vehicles are assigned to the same depot, then there is a high risk of inconsistent assignment. In spite of this identified drawback, we have verified that Greedy1 efficiently and fastly assigns requests on each vehicle.

Greedy2 algorithm

As we explained above, the assignment strategy followed by Greedy1 is based on distance and time indicators. But it does not consider parameters like the vehicle load. Thus the resulting solutions can be fairly unbalanced regarding load distribution among vehicles. Considering this potential drawback we have developed a second heuristic algorithm called Greedy2, with complexity also in $O(n^2)$, in which we consider the theoretical average load to assign to each vehicle so it satisfies all requests. This one is weighted with a fixed rate (ltm) to define a maximal load allowed per vehicle (labelled σ) which involves a change of vehicle in the assignment procedure as soon as it is reached or exceeded. The resulting value may be seen as a threshold aiming at balancing the load between vehicles. The greater the rate ltm is, the more unbalanced assignments are allowed.

σ = $(\sum_{r \in R} q_r \ / \ |V|) \cdot ltm$

Random assignment

This algorithm randomly chooses one vehicle v in B_r for every demand $r \in R$. We already ensured that the output of this algorithm leads to feasible solution because delays on any time window are allowed. If the capacity of the assigned vehicle does not allow multiple requests to be loaded at the same time then one solution is still to pick up and to deliver requests one after another.

Generation of initial assignment solutions

Fig.2 explains how the three above heuristics are used to generate a set of initial solutions solving the assignment problem. Each one contributes equally to this generation and provides one third of the population. At the end of this step, the redundant solutions are erased to keep distinct ones.

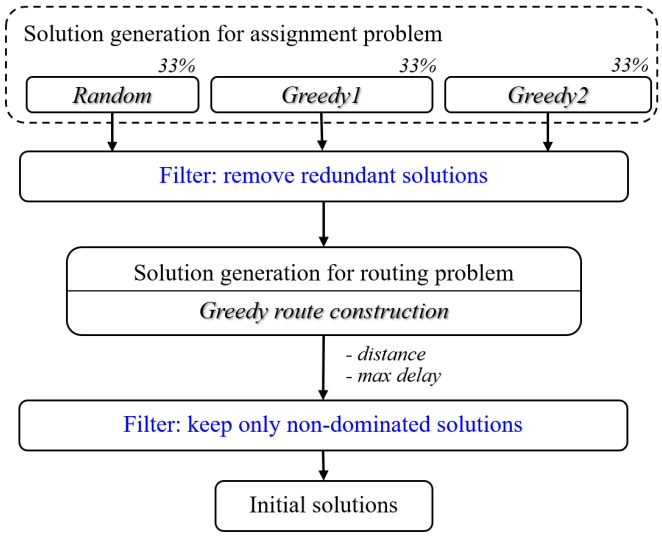

Fig. 2. Generation of initial solutions with stage 1

The interest of combining the three heuristic procedures is illustrated through the figures 3, 4 and 5 for the two instances R01 and R04 (with respectively 3 and 12 vehicles, 30 and 150 demands, 2 and 4 transfer points). Indeed, we can observe that greedy1 algorithm rather tends to fill the vehicles as more as possible, then it may involve better distance but more time window violations. Greedy2 algorithm tends to balance the load of the vehicles, which may increase the traveled distance but then may limit the delays. If for R03, the two research sub-spaces explored with those greedy procedures may overlap, they are obviously distinct for R04, which justifies their combined use. As for random assignment algorithm, it is used to ensure not to forget some areas of the global search space. Its efficiency depends on the studied instance and on the size of this one. It is shown in figure 3 where the best distance is obtained with this random procedure at stage 1.

2) Routing problem: At this step, each demand $r \in R$ is assigned to a vehicle $v \in V$. Thus it is possible to build a route for each vehicle independently. Here we must solve a Pickup & Delivery Problem with Time Windows and Paired Demands (PDPTWPD). We consider for any $v \in V$ a set O_v containing pickup and delivery operations op assigned to the vehicle. For each vehicle the assigned operations are progressively inserted using a nearest neighborhood strategy, where the next operation op, added at the end of

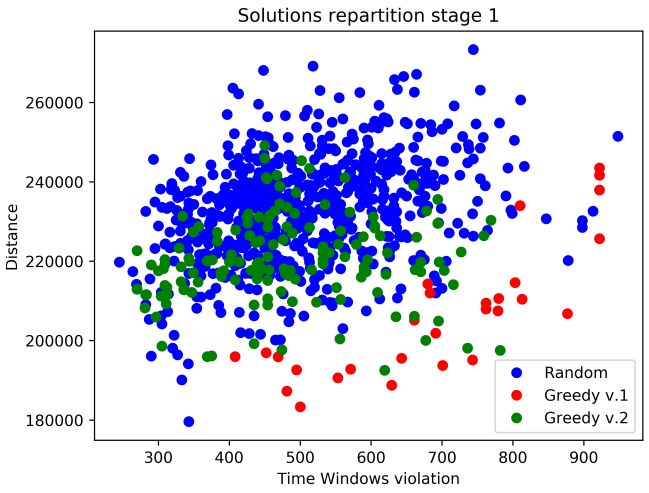

Fig. 3. Initial solutions for instance R01 after stage 1

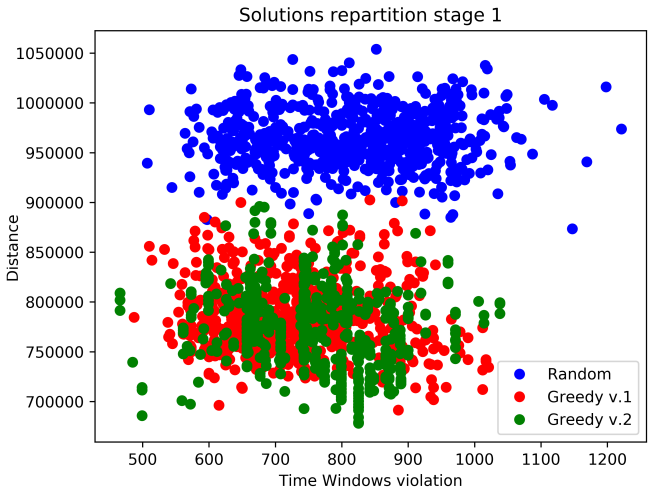

Fig. 4. Initial solutions for instance R03 after stage 1

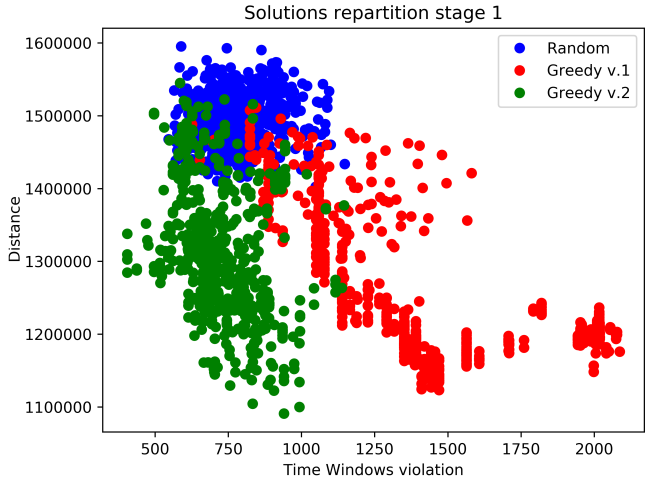

Fig. 5. Initial solutions for instance R04 after stage 1

the partial route is the one with the best minimal score $GS_v^{'op}$. $GS_v^{'op}$ is dynamically computed in the same way as GS_v^r used in Greedy1 and Greedy2, but by distinguishing pick up and delivery operations. Three weights $wps^{'}$, $wtss^{'}$ and $wtrs^{'}$ are also submitted to the algorithm. We define the computation of $GS_v^{'op}$ as follows:

$PS_v^{'op}$ Physical proximity indicator (lower is better)

$TSS_v^{'op}$ Violation of the operation site time window if vehicle v visits it (higher is prioritized) and the origin/destination sites (higher is better)

$TRS_v^{'op}$ Violation of the demand time window if vehicle v visits it (higher is prioritized)

L_v^{op} Load score used to forbid the insertion of op if the resulting load of vehicle v exceeds its capacity Q_v after loading ($= +\infty$). Else insertion is allowed ($= 0$)

$GS_v^{'op}$ $= L_v^{op} + PS_v^{'op} \cdot wps/maxd$ $-(TSS_v^{'op} \cdot wtss + TRS_v^{'op} \cdot wtrs)\ /\ maxt$ Concatenated score for pair $(op,\ v)$

Weights used in stage 1 are balanced, i.e. $wps^{'}$, $wtss^{'}$ and $wtrs^{'}$ have the same value. At the end of stage 1, the obtained solutions (assignment and routes) are filtered to keep only non dominated solutions. Then we extract a Pareto front of the explored sub-space. It constitutes our initial set of solutions, to be improved in the two other stages.

B. Stage 2

For each solution of the initial set, we apply an iterated local search using the principles of the above routing algorithm. At a given iteration, we generate eight neighbors by varying the values of the last defined weights (then modifying the routes), with a given increment I. Two neighbors are generated by randomly increasing two weights. Others result from the increase (2 neighbors) or decrease (2 neighbors) of a single weight randomly chosen; the last two neighbors are obtained by randomly and simultaneously increasing one weight and decreasing another one. If all neighbors are dominated by the current solution, then this one is kept at the next iteration. If one or more neighbors are not dominated by the current solution (better for at least one criterion), then one of them is randomly chosen for the next iteration. The number of iterations is a priori fixed and constitutes the stopping criterion. At the end of stage 2, each initial solution (cross markers in Fig.6) has evolved towards a new one through intermediate solutions (dot markers in Fig.6). Different colours are used to keep track of the initial solution during evolution process. This new set of resulting solutions is an evolution of the initial Pareto front. We can see that this stage enables us to improve substantially the quality of the solutions in terms of distance and delays.

C. Stage 3

After improving the distance and the delays for the initial defined assignments, we operate another improvement leverage by allowing transshipment which makes the satisfaction

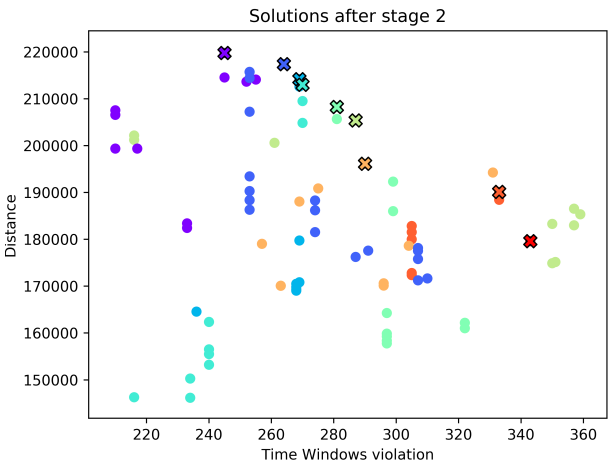

Fig. 6. Solutions obtained for instance R01 after stage 2

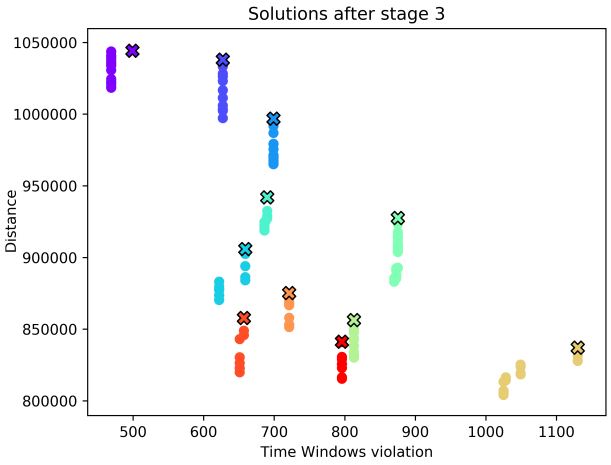

Fig. 8. Evolution of solutions obtained in stage 2 on instance R04

of demands into a multimodal problem. Stage 3 applies a transfer insertion heuristics. For a given solution, and for each possible couple of vehicles (v1, v2), we determine the common covering area of this couple (see figure 7). If a transfer site is located in this area, and for each demand of v1 whose the destination belongs to this zone, the best transfer insertion cost is computed. The insertion consists in testing all the possible insertion positions of delivery at the new site (transfer site) for v1 (after the associated pick up operation of course); for v2, the routing algorithm is used to insert both pick up at transfer point and delivery at destination of the demand newly assigned. Fig.8 illustrates the potential improvement level provided by this stage.

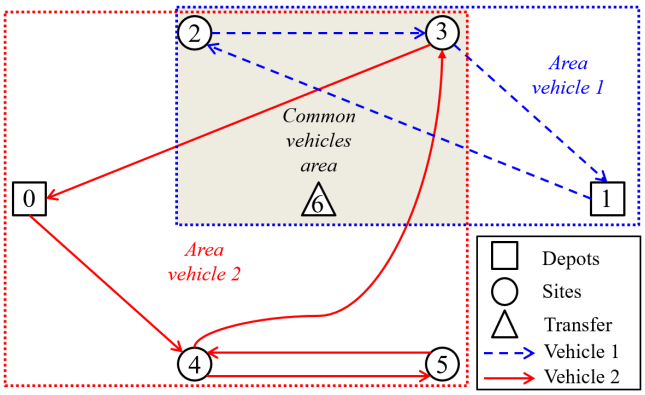

Fig. 7. Covering areas of two vehicles and their intersection where gains on transfer insertions are computed

V. Computational results

Algorithms are developed under Python 2.7 on a Intel™ i7-6700HQ running on a single thread at 2.6 GHz. In stage 1, 2000 initial solutions were created, evenly distributed among the 3 strategies (fig.2). In stage 2, we applied 100 iterations for local search on each solution. Variation step for greedy weights wpd', wpd' and wpd' is set to 20%. Each weight was initialized in stage 1 with value $1/3$. These weights are normalized each time their value varies. In stage 3, we set 500 iterations for transfer insertion. If no gain was possible at a given iteration, then we stopped the optimization process. In order to assess our algorithm we have not found in the literature instances gathering all the characteristics of our studied problem (multiple vehicle depots, heterogeneous fleet of vehicles, paired demands, TW on sites, TW on demands, transfers between vehicles and so on). So we have generated new instances. Table I provides a synthesis of the results we obtained by running our algorithm on 9 among those generated instances (with different sizes and configurations). We got reference results for small size instances (optimal for F01 to F03, and an upper bound for F04), by applying a previously developed exact method (see Godart et al. [4]). However it was not the case for bigger size instances (R0i) as the exact methods failed to solve them in reasonable time. Table I gives the couple (distance, delay) for two particular solutions among those obtained: solution β with the minimal found distance and solution α with the minimal found delay (smallest maximal tardiness). But our algorithm works on a population of solutions which are fast generated and improved. Indeed, for small instances, the CPU time is about 10 seconds. It varies from about one minute to less than one hour for the biggest instances of Table I. The table provides the part of the total CPU time for each stage of the algorithm, which varies according to the size of the instances and in particular the number of transfer insertions. For small instances, the best known results (BKS) give the best distance with no tardiness in demand and site time windows. We also found solutions with no delay for almost all the small instances, and we also provide solutions with better distances with some delays. Then we provide potentially flexible solutions, as such a trade-off may interest a carrier, on condition that he may negotiate with his customers to deliver goods not exactly on time. For the big instances, which are much more complex and constrained, the obtained solutions are associated with delays (however sometimes small like for R02). Nevertheless, missing reference results, it is difficult to know if there exist or not solutions with no delay, knowing that: there is a limited amount of vehicles,

			F00	F01	F02	F03	F04	R01	R02	R03	R04
	Instance configuration	vehicles	**2**	**2**	**2**	**3**	**4**	**3**	**12**	**20**	**12**
		demands	**3**	**4**	**6**	**4**	**8**	**30**	**50**	**100**	**150**
		transfer points	**1**	**1**	**1**	**1**	**2**	**2**	**6**	**6**	**4**
Exact	**BKS** (with no delay)	Distance value	**3532**	**3437**	**3828**	**2979**	**6478**	-	-	-	-
		Optimality gap	-	-	-	-	**14**	-	-	-	-
		CPU (s)	**2.6**	**2.6**	**3187**	**133**	**1209600**	-	-	-	-
Greedy	CPU (s)	Stage 1	6.3	5.8	4.8	6.4	8.3	16.9	32.5	83.0	128.4
		Stage 2	2.3	2.4	3.8	2.5	4.0	40.2	45.1	72.6	458.4
		Stage 3	0	0	0	0.02	0.13	20.6	137.2	344.5	2311.2
		Total	8.6	8.2	8.6	8.9	12.5	77.6	214.8	500.1	2897.9
	Stage 1: Distinct solutions count		8	16	64	81	675	842	1597	1889	1875
	Stage 1: Non-dominated solutions count		4	3	4	3	3	9	9	4	11
	Stage 2: Local improvements count		0	0	0	0	3	78	153	73	377
	Stage 3: Transfer insertions count		0	0	0	0	1	7	49	58	142
	Solution **β** with min distance	Distance	3247	3590	2907	2377	6056	146298	316558	587266	804141
		Delay	35	0	48	8	28	216	277	879	1025
	Solution **α** with min delay	Distance	3596	3590	3939	3096	7501	173519	371359	650820	1018475
		Delay	2	0	0	0	0	138	22	473	469
	Mean improvement	Distance	0	0	0	0	-306	-37867	-17179	-86799	-284770
		Delay	0	0	0	0	-2	-30	32	58	46
	Min/Max improvement	Distance	0/0	0/0	0/0	0/0	-25/ -642	-6074/ -66669	-2933/ -32951	-24465/ -143345	-242230/ -340718
		Delay	0/0	0/0	0/0	0/0	0/ -6	63/ -205	139/ -10	196/ -12	321/ -182
	Distance gap between sol. **α** and **BKS**		1.8%	4.5%	2.9%	3.9%	15.8%	-	-	-	-
	Distance gap between sol. **β** and **BKS**		-8.1%	4.5%	-24.1%	-20%	-6.5%	-	-	-	-

TABLE I

COMPUTATIONAL RESULTS OBTAINED ON 9 INSTANCES

some of these vehicles are not allowed to deliver some of the requests (set B_r), and the time windows are rather tightened on sites, and even more on demands.

Finally it is not so easy to qualify the improvement obtained with our approach. However, between the solutions α and β, we can observe variations from 11% to 35% for the distance and from 36% to 100% for the delay. Besides, the bigger the mean number of demands per vehicle is, the higher the improvement potential is. We can suppose that it comes from the great number of possible assignments of requests to the vehicles, and also from the great number of possible permutations within each route. Nevertheless, it let us hope a strong efficiency of our algorithm for even larger instances.

VI. CONCLUSION

We have proposed an algorithm combining several heuristics working on a population of solutions. This method has provided encouraging results showing its efficiency to solve a bi-objective variant of Pick up and Delivery problem with transfers. The greedy strategy provides good solutions within acceptable computation time. Further extensions may be envisaged for this work, like solving large scale instances and comparing solution metrics with mono-objective algorithms. We could also adapt this method to a dynamic variant of the problem (dynamic insertion of new requests). In this case, the assignment procedure Greedy2 would start with current load on each vehicle, according to previous scheduling. Routing procedure would become an insertion procedure considering the existing routes. Stage 3 would then be used as it is.

REFERENCES

[1] B. Coltin and M. Veloso, Scheduling for Transfers in Pickup and Delivery Problems with Very Large Neighborhood Search, Twenty-Eighth AAAI Conf. Artif. Intell., pp. 2250-2256, 2014.

[2] S. Deleplanque and A. Quilliot, Dial-a-ride problem with time windows, transshipments, and dynamic transfer points, vol. 46, no. 9. IFAC, 2013.

[3] S. Gao, Y. Wang, J. Cheng, Y. Inazumi, and Z. Tang, Ant colony optimization with clustering for solving the dynamic location routing problem, Appl. Math. Comput., vol. 285, pp. 149-173, Jul. 2016.

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

[5] I. Karaoglan, F. Altiparmak, I. Kara, and B. Dengiz, The location-routing problem with simultaneous pickup and delivery: Formulations and a heuristic approach, Omega, vol. 40, no. 4, pp. 465-477, 2012.

[6] R. Masson, F. Lehuédé, and O. Péton, An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers, Transp. Sci., vol. 47, no. 3, pp. 344-355, 2013.

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

[8] Y. Zare Mehrjerdi and A. Nadizadeh, Using greedy clustering method to solve capacitated location-routing problem with fuzzy demands, Eur. J. Oper. Res., vol. 229, no. 1, pp. 75-84, Aug. 2013.

[9] S. Mitrović-Minić and G. Laporte, The pickup and delivery problem with time windows and transshipment, INFOR, vol. 44, no. 3, pp. 217-227, 2006.

[10] G. Nagy and S. Salhi, Location-routing: Issues, models and methods, Eur. J. Oper. Res., vol. 177, no. 2, pp. 649-672, Mar. 2007.

[11] A. Spickermann, V. Grienitz, and H. A. von der Gracht, Heading towards a multimodal city of the future?, Technol. Forecast. Soc. Change, vol. 89, pp. 201-221, 2013.

[12] A. Godart, H. Manier, C. Bloch, and M. A. Manier, Hybrid metaheuristic for the Pickup and Delivery Problem designed for passengers and goods transportation, IFAC MIM 2019 Conf. (Berlin, Germany).