INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 3, PP. 1–9
Manuscript received February 7, 2025; revised June 2025. doi: 10.24425/ijet.2025.153624

Multi-criteria vehicle routing problem for a real-life parcel locker-based delivery

Radosław Idzikowski, Jarosław Rudy, and Michał Jaroszczuk

***Abstract*—In this paper, a multi-criteria Vehicle Routing Problem with distance and capacity constraints for modeling a delivery system with parcel locker, is considered. The problem is formulated and two optimization criteria are defined. The first criterion minimizes the total travel time of all vehicles and the second criterion minimizes the total penalty for late delivery of orders. Three solving methods, relying on the concept of Pareto-optimality, are proposed: a greedy constructive heuristic, a Tabu Search metaheuristic and a Genetic Algorithm. A number of benchmark instances are created using real-life parcel locker locations and traveling times, from one of the major cities in Poland. In preliminary research, sorting strategies for the greedy method are tested, with the sorting based on deadline–arrival difference to priority ratio yielding the best performance in all tested cases. Next, computer experiments are performed to evaluate the quality of the proposed methods, using the concept of Hypervolume Indicator. Results confirm that both Tabu Search and Genetic Algorithm significantly improve the solution provided by the greedy algorithm, with Genetic Algorithm being the most effective on average. However, results also indicate that both Tabu Search and Genetic Algorithm have different effectiveness in different cases. It is concluded that the best performance is achieved by both algorithms being used in parallel, complementing each other.**

***Keywords*—Vehicle Routing Problem; parcel lockers; multi-criteria optimization; operations research; Pareto optimality; metaheuristics**

I. Introduction

VEHICLE Routing Problems (VRP) is one of the classic problems in combinatorial optimization. It is well-known due to its difficulty (VRP is NP-hard), as well as its various practical applications which range from more mundane such as waste management [1] and supply chains [2] through more unusual like biomass transport [3] and cash transit [4] to post-disaster delivery of water [5]. Thus, the topic of VRP has received considerable attention from both the industry and researchers, with a great number of papers published every year. This resulted in a large number of different VRP variations emerging in the literature.

Radosław Idzikowski is with Department of Control Systems and Mechatronics, Wrocław University of Science and Technology, Poland (e-mail: radoslaw.idzikowski@pwr.edu.pl, ORCiD: 0000-0001-5232-1998).

Jarosław Rudy is with Department of Control Systems and Mechatronics, Wrocław University of Science and Technology, Poland (e-mail: jaroslaw.rudy@pwr.edu.pl, ORCiD: 0000-0003-1095-6041).

Michał Jaroszczuk is with Department of Control Systems and Mechatronics, Wrocław University of Science and Technology, Poland (e-mail: michal.jaroszczuk@pwr.edu.pl, ORCiD: 0009-0009-8617-889X).

One of the possible applications of VRP systems in modern world is delivery service with the use of parcel lockers. Instead of delivering packages directly to customers and risking their absence, the delivery is made to a smaller number of parcel lockers with fixed locations. This allows the customers to pick up their parcels at their leisure.

While convenient for the customers, such a delivery service has several characteristics that need to be taken into account. First, vehicle transporting the parcels have limited capacity (especially if small cars are employed). Second, travel time of a vehicle is limited due to working hours of the drivers (e.g. 8 hours a day). Third, parcels might arrive at the central depot at different times, making a vehicle unable to start making deliveries, until all of the parcels assigned to it have arrived and have been loaded. Fourth, multiple orders commonly have the same parcel lockers as their destination. In a regular VRP this is can be represented by merging such multiple orders into one. However in our case, this becomes more complex due to aforementioned arrival time as parcels headed to the same locker might become available at different times. As such, it might be beneficial to deliver them via different vehicles at different times of the day.

In such a delivery system there are two crucial aspects that should be optimized. First, is the minimization of vehicle operations costs (e.g. fuel, salary of drivers). Second, is minimization of late delivery of packages. Nowadays, customers are informed of estimated delivery times ahead of time. As such, late deliveries reflect poorly on the company, especially if the customer already waited a long time (e.g. for the parcel to be shipped from overseas).

In this paper, we propose an optimization approach to the aforementioned delivery system with parcel lockers, modeled as a specific VRP variant. The main contributions of this paper are as follows:

1) A problem of modern delivery service using parcel lockers is formulated and modeled as a variant of a Vehicle Routing Problem.
2) We consider simultaneous minimization of two criteria: total time of delivering packages and total penalty for late deliveries.
3) We propose two metaheuristic solving methods—Tabu Search (TS) and Genetic Algorithm (GA)—capable of tackling problem instances of practical size in short running time, as well as a greedy heuristic method serving as the baseline.

4) We propose the sorting strategy for the greedy method, based on 243 strategies tested in preliminary research.
5) We perform computer experiments using instances based on real-life data from one of major cities in Poland.
6) We highlight the strengths and limitations of the proposed solving methods and show the synergy achieved from applying both of them at the same time.

The remainder of this paper is organized as follows. Section II presents the literature overview. In Section III the problem is formulated and matters regarding the goal function calculation and solution feasibility are discussed. Section IV describes the proposed solving methods. Section V presents the result of numerical experiments. Finally, Section VI contains the conclusions.

II. Literature review

VRP with time travel minimization as well as time and capacity constraints is a known optimization problem, with many existing research and extensions being considered due to their practical applications. Results of some of such works, can be compared to the multi-criteria parcel locker delivery VRP, presented in this paper. However, to our best knowledge, there is only a few articles which focus on multi-objective VRP, where the criteria are the total travel time and penalty for late deliveries. We start by discussing several such papers.

A similar VRP variant was examined by Wen et al. [6], where optimization of delivery cost to self pick-up lockers, was being compared to a traditional home delivery. The optimization criteria were based on distance, penalty for exceeding time windows and CO_2 emission costs. On the other hand, Abdullahi et al. [7] formulated a multi-criteria VRP with the optimization of total transportation costs, environmental factor and social aspects, but without considering the maximal vehicle travel time and order deadlines. In their experiments, the proposed Biased-Radomized Iterated Greedy Local Search is compared to the best known solutions of adapted VRP instances. Similarly, environmental factor was one of the criterion in Gulmez et al. [8] paper. The authors considered fuel/electricity limitation, instead of travel time constraint. Four mainstream solving methods are tested in the experiments with NSGA-II obtaining the best performance. NSGA-II is also the chosen solving method by Srivastava et al. [9], where 5 objectives, including the number of vehicles, delay time and total travel distance, are considered and compared with existing literature results. Finally, bi-criteria optimization of the total travel time and the number of vehicles was investigated by Duan et al. [10]. Differently than in many other approaches, uncertainty of the travel time and time windows was considered as well. As solving method, a robust multi-objective Particle Swarm Optimization was proposed.

We will now focus on existing VRP approaches that are similar to our research, but which do not consider multi-criteria optimization. A problem of delivery to parcel lockers with travel time and vehicle capacity constraints was studied by Orenstein et al. [11]. The authors presented a formal Mixed-Integer Linear Programming formulation and proposed a TS algorithm. Similar VRP variants with time and capacity constraints were considered by Li et al. [12] and Pan et al. [13]. Both papers used Adaptive Large Neighborhood Search to solve the formulated problem, having such properties as time windows, ready times and service time assigned to each order. However, the problems in those papers were not applied to the concept of parcel lockers delivery. An interesting work was done by Cokyasar et al. [14], where a case study of a TCVRP variant was examined using benchmarks based on data from biggest delivery companies in the US. In the paper, the impact of maximal travel time, vehicle capacity, service time and vehicle range were analyzed, with goal of minimizing the total travel distance. Finally, Grabenschweiger et al. [15] considered a time- and capacity-constrained VRP with delivery to heterogeneous locker boxes, but without taking into account order ready time and maximal vehicle capacity.

To summarize, the topic of VRP modeling parcel lockers-based delivery is not well-researched in the literature. This is especially true when multi-criteria goal of optimizing total travel time and total late deliveries penalty and other problem constraints are considered. The closest the aforementioned paper [6], but even that differs significantly from our approach with regards to the assumptions. Furthermore, existing approaches very rarely conduct experiments using real-life data. As such, we address those shortcomings in our paper.

III. Problem formulation

The VRP variant considered in this paper is formulated as an modification to the model shown in [16]. Since the full mathematical formulation is cumbersome, we will omit it and use the Giant Tour Representation (GTR) instead. The summary of the notation is shown in Table I.

A. Input data

We are given a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of n delivery orders and a set $\mathcal{L} = \{0, 1, 2, \ldots, l\}$ of $l + 1$ locations. Locations 1 through l are delivery locations (i.e. parcel lockers), while location 0 is the depot. For each order $i \in \mathcal{N}$ we define its arrival time $a_i > 0$, deadline (due-date) $d_i > a_i$, goal (destination) location $g_i \in \mathcal{L} \setminus \{0\}$, weight (mass) $w_i > 0$ and priority $p_i > 0$. By $S > 0$ we denote the constant time required to service a single order (retrieving the package, placing it in the package locker etc.). Moreover, for each ordered pair of locations $k, l \in \mathcal{L}$ we denote the travel time between them as $t_{k,l} \geq 0$. Due to location describing points in a city, it is usually expected that $t_{k,l} \neq t_{l,k}$. Naturally, $t_{k,k} = 0$ for all k. Next, we are given a set $\mathcal{V} = \{1, 2, \ldots, v\}$ of $v \leq n$ identical vehicles. By $C > 0$, $T > 0$, and $P > 0$ we denote the vehicle capacity, the time limit for vehicle travel (e.g. daily driver worktime limit) and the time needed for the vehicle to park and take off from a location respectively.

B. Representation

In order to formally define the goal function or the constraints, we will represent the solution using a modified GTR. A regular GTR is a sequence of all orders/locations with a number of separating zeroes (indicating change of vehicle),

TABLE I
NOTATION

Symbol	Meaning
	Indexes
i, j	general indexes and indexes for orders
k, l	indexes for locations
r	index for vehicles
	Input data
$\mathcal{N}$, n	sets of orders, number of orders
$\mathcal{L}$, l	sets of locations, number of locations
$\mathcal{V}$, v	sets of vehicles, number of vehicles
a_i	arrival time of order i
d_i	deadline (due-date) of order i
g_i	goal (destination) of order i
w_i	weight (mass) of order i
p_i	priority of order i
$t_{k,l}$	travel time from location k to location l
S	order servicing time
P	time needed for vehicle to park and take off
T	time limit for vehicle travel
C	vehicle capacity
	Solution and its derivatives
π	solution
π_r	sequence of orders assigned to vehicle r according to π
σ_r	sequence of order destinations according to π_r
ρ_r	sequence of parking times according to π_r
$A_r(\pi)$	ready time for vehicle r according to π
$D_{i,r}(\pi)$	delivery time of the i-th order in vehicle r according to π
	Other notation
$\text{len}(x)$	length (number of elements) of sequence x
$x(i)$	i-the element of sequence x
$i \in x$	i is an element of sequence x

mixed in with the orders. In our case, orders are not equivalent to locations (e.g. there can be multiple orders per location and there can be locations with no orders), so we will modify how GTR is interpreted.

Let us consider example shown in Figure 1. Here we have 9 locations (A to I), 10 orders (1 to 10) and 3 vehicles. In this example, vehicle 1 first visits location F to deliver order 8, then moves to location G to deliver order 5, then delivers order 1 at location I and finally returns to the depot. For such an example the corresponding GTR solution π is:

$$(8, 5, 1, 0, 9, 6, 10, 0, 4, 2, 3, 7). \tag{1}$$

It should be noted that π stores orders and vehicle separators only. Locations are not stored in π explicitly. Different solutions are obtained by re-arranging the elements of π.

We will now derive from π a few symbols to simplify the notation. By π_r, $r \in \mathcal{V}$ we denote the subsequence of π containing orders assigned to vehicle r. By σ_r we denote a sequence of destinations of orders from π_r i.e. $\sigma_r(i) = g_{\pi_r(i)}$. Similarly, by ρ_r we denote sequence indicating parking times associated with orders in π_r. Parking time is P if the previous destination is different and 0 otherwise:

$$\rho_r(i) = \begin{cases} P & \text{if } i = 1 \vee \sigma_r(i) \neq \sigma_r(i-1), \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

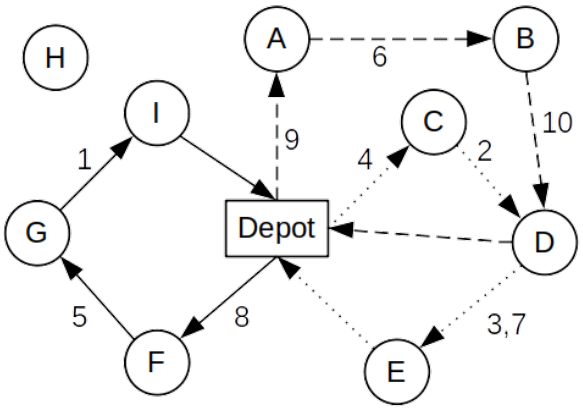

Fig. 1. Example tours for 10 orders, 9 locations and 3 vehicles

For the example solution in (1) we have:

$$\pi_1 = (8, 5, 1) \quad \pi_2 = (9, 6, 10) \quad \pi_3 = (4, 2, 3, 7) \tag{3}$$
$$\sigma_1 = (F, G, I) \quad \sigma_2 = (A, B, D) \quad \sigma_3 = (C, D, E, E) \tag{4}$$
$$\rho_1 = (P, P, P) \quad \rho_2 = (P, P, P) \quad \rho_3 = (P, P, P, 0) \tag{5}$$

C. Goal function

The goal is to find solution π^* which minimizes a given two-criteria goal function $F(\pi)$:

$$F(\pi^*) = \min_{\pi} F(\pi). \tag{6}$$

$F(\pi)$ consists of two criteria $F_1(\pi)$ and $F_2(\pi)$. Criterion $F_1(\pi)$ is the total time taken for all vehicles to deliver all orders. For a given π this criterion is expressed as follows:

$$F_1(\pi) = \sum_{r \in \mathcal{V}} \left(t_{0,\sigma_r(1)} + t_{\sigma_r(\text{len}(\sigma_r)),0} + \sum_{i=2}^{\text{len}(\sigma_r)} t_{\sigma_r(i-1),\sigma_r(i)} + \sum_{i=1}^{\text{len}(\pi_r)} \rho_r(i) \right) + nS, \tag{7}$$

Formula (7) for each vehicle includes 4 terms:

1) $t_{0,\sigma_r(1)}$ – time vehicle takes to travel from the depot to the first location assigned to it.
2) $t_{\sigma_r(\text{len}(\sigma_r)),0}$ – time vehicle takes to travel from its last assigned location back to the depot.
3) $\sum_{i=2}^{\text{len}(\sigma_r)} t_{\sigma_r(i-1),\sigma_r(i)}$ – time taken for the vehicle to travel through its assigned sequence of locations.
4) $\sum_{i=1}^{\text{len}(\pi_r)} \rho_r(i)$ – the time vehicle takes while parking.

This form of function F_1 was chosen to directly or indirectly optimize 1) fuel-usage, 2) man-hours and 3) number of vehicles and their utilization.

The second criterion $F_2(\pi)$ is the total penalty for delivering orders past their deadline, which is expressed as follows:

$$F_2(\pi) = \sum_{r \in \mathcal{V}} \sum_{i=1}^{\text{len}(\pi_r)} \max\{0, D_{i,r}(\pi) - d_{\pi_r(i)}\} \cdot p_{\pi_r(i)}, \tag{8}$$

where $D_{i,r}(\pi)$ is the actual time the i-th order in vehicle r is delivered:

$$D_{i,r}(\pi) = A_r(\pi) + t_{0,\sigma_r(1)} + \sum_{j=2}^{i} t_{\sigma_r(j-1),\sigma_r(j)} + \\ + \sum_{j=1}^{i} \rho_r(j) + iS. \tag{9}$$

Here $A_r(\pi)$ is the time vehicle r can starts its tour i.e. the time all of its orders have arrived in the depot:

$$A_r(\pi) = \max_{i \in \pi_r} a_i. \tag{10}$$

Formula (9) includes 5 terms:

1) $A_r(\pi)$ – time vehicle waits before it leaves the depot.
2) $t_{0,\sigma_r(1)}$ – time vehicle takes to travel from the depot to the first location assigned to it.
3) $\sum_{j=2}^{i} t_{\sigma_r(j-1),\sigma_r(j)}$ – time taken for the vehicle to travel through its tour until (and including) its i-th order.
4) $\sum_{j=1}^{i} \rho_r(j)$ – the time vehicle takes while parking during its tour until (and including) its i-th order.
5) iS – the time to service the first i orders of the vehicle.

D. Constraints

The goal function F has to be minimized under the following constraints:

1) Each order is assigned to exactly one vehicle.
2) Each vehicle with at least one order, starts its tour at the depot and has to return to the depot after completing its tour.
3) Each vehicle can only leave the depot after all orders assigned to it have arrived.
4) Capacity for any vehicle does not exceed C.
5) Total travel time for any vehicle does not exceed T.

Constraints 1) and 2) are ensured by GTR itself. Constraint 3) is already accounted in formulas (7) and (9). Constraint 4) implies that for all $r \in \mathcal{V}$:

$$\sum_{i \in \pi_r} w_i \leq C. \tag{11}$$

Finally, constraint 5) for each vehicle $r \in \mathcal{V}$ sums the travel, parking and service time for all orders assigned to r:

$$t_{0,\sigma_r(1)} + t_{\sigma_r(\mathrm{len}(\sigma_r)),0} + \sum_{j=2}^{\mathrm{len}(\sigma_r)} t_{\sigma_r(j-1),\sigma_r(j)} + \\ + \sum_{j=1}^{\mathrm{len}(\rho_r)} \rho_r(j) + \mathrm{len}(\pi_r)S \leq T. \tag{12}$$

E. Solution value and feasibility

Despite the complexity of the formulation, the determination of a goal function value and feasibility for a given π can be done in a linear time using the following procedure. We first initialize necessary variables (empty list of orders, zero current time, zero vehicle load, zero latest arrival time). We then start to scan the solution from $\pi(1)$ to $\pi(\mathrm{len}(\pi))$. On encountering orders, we add them to the list, add order weight to vehicle load and update the latest (highest) arrival time.

On encountering the end of the current vehicle (i.e. a zero) we try to „complete" the vehicle. If the vehicle load exceeds its capacity C, the solution is infeasible and the procedure ends. Otherwise, we set the current time to the latest arrival order (this cannot be done earlier as we need to know arrival times for all orders of this vehicle to know when the vehicle can leave the depot). We then iterate over the orders collected in the list, advancing time as required (traveling to the next location, parking, servicing order). For each order we calculate the tardiness penalty for the order (which is based on the current time) and add it to the second criterion value. After all orders of the vehicle have been processed, we check if the overall vehicle time (counted from the moment it leaves the depot) exceeds the time limit T. If it does, then the solution is infeasible and the procedure ends. Otherwise, the vehicle time (including waiting for all of its orders to arrive) is added to the first criterion value. After all vehicles have been processed, the procedure ends, returning values of both criteria. As can be easily seen, the total running time of this procedure is $O(n + v) = O(n)$, since $v \leq n$.

The procedure itself checks for the violation of constraints 4) and 5), while also ensuring that constraint 3) is met. Constraints 1)–2) are not checked explicitly, however, they are always met as long as GTR is properly formed. If GTR is not well-formed then other situations (e.g. exceeding the allowed numbers of vehicles, use of non-existent orders) are possible as well. In our case, we assume that the solving methods will operate on well-formed GTRs only. However if necessary, the appropriate checks could be easily added, while not affecting the $O(n)$ running time complexity.

IV. Solving methods

Due to large problem sizes encountered in practice (hundreds or thousands of orders), the exact solving methods are inapplicable in real-life parcel delivery. Thus, in this section we propose three heuristic solving methods for the considered problem: (1) a greedy constructive method, (2) Tabu Search method and (3) Genetic Algorithm.

A. Greedy heuristic

The greedy heuristic starts by creating "empty" solution π that contains only empty vehicles:

$$\pi = (\underbrace{0, 0, \ldots, 0}_{v-1 \text{ times}}). \tag{13}$$

Next, the set of orders $\mathcal{N}$ is sorted according to a chosen strategy. After this initial phase, the algorithm proceeds iteratively. In each iteration we try to insert the next order (as according to the chosen sorting strategy) into the partial GTR. We tentatively insert the order into all available positions and evaluate the goal function for each possibility. The order is ultimately inserted into the position that yields the best result. However, we need to choose one insert position and the goal function is two-criteria. Thus, we need a way to be able to

compare each result. A well-known TOPSIS [17] method, able to rank multi-criteria solutions is used for this.

The last iteration proceeds differently. Since only one order remains to be inserted, each insertion will result in a complete GTR. Thus, we consider and evaluate all possible insertion points and collect them into one set. Solutions that are dominated (according to the Pareto-optymality principle) are removed from the set. The remaining non-dominated solutions are returned by the method. Since $v \leq n$, the total running time of the greedy method is $O(n^3)$.

B. Tabu Search

Tabu Search (TS) is a well-known local-search metaheuristic. It has been successfully applied for solving a wide range of optimization problems including VRP [11], scheduling [18] and bin packing [19]. TS starts from an initial solution and tries to iteratively improve it, until a chosen stopping condition is met. In each iteration a neighborhood of the current solution is searched and the best found solution replaces the current one. In order to avoid re-visiting past solutions, the algorithm uses short-term memory, called the tabu list, to mark certain solutions as forbidden.

In our case, the initial Pareto front $\mathcal{P}$ is provided by the greedy algorithm from the previous subsection. Next, TOPSIS method is used to choose a single solution from $\mathcal{P}$ that will act as our current solution π. In our implementation the move consists of swapping elements in π i.e. neighboring solution π' is created by swapping $\pi(i)$ and $\pi(j)$ for some i and j. Since swapping $\pi(i)$ with $\pi(j)$ is the same as swapping $\pi(j)$ with $\pi(i)$ we assume that $j > i$. Also, swapping when $i = j$ does not change the solution and is thus ignored. The size of the resulting neighborhood is:

$$\frac{\mathrm{len}(\pi)(\mathrm{len}(\pi)-1)}{2} \in O(\mathrm{len}(\pi)^2) = O(n^2). \qquad (14)$$

The TS method considers all neighbors π' as follows. Neighbors created from the moves that are currently forbidden are ignored. The rest of the neighbors is evaluated and checked for feasibility. Infeasible neighbors are also ignored. The remaining (i.e. feasible and not forbidden) neighbors create a separate Pareto set of candidates $\mathcal{C}$. TOPSIS method is then used to select from $\mathcal{C}$ the best candidate π_c. Next, we try to update the Pareto front $\mathcal{P}$ with π_c. Namely, π_c is added to $\mathcal{P}$ if π_c is not dominated by any solution in $\mathcal{P}$. Similarly, all solutions in $\mathcal{P}$ dominated by π_c are removed from $\mathcal{P}$. Finally, π_c becomes the current solution π for the next iteration.

The tabu list is in the form of a matrix M of size $\mathrm{len}(\pi) \times \mathrm{len}(\pi)$. Element $M_{i,j} = z$ indicates that the move swapping $\pi(i)$ and $\pi(j)$ is forbidden up until iteration z. In the beginning $M_{i,j} = 0$ (i.e. all moves are allowed). Every time a best candidate π_c is chosen, we forbid the move that led from π to π_c by setting $M_{i,j}$ to $\mathrm{it} + \mathrm{cad}$, where it is the current iteration number and cad is the number of iteration for which the move will be forbidden (i.e. cadence). In our implementation we set:

$$\mathrm{cad} = \left\lfloor \sqrt{\mathrm{len}(\pi)} \right\rfloor = \left\lfloor \sqrt{n + c - 1} \right\rfloor . \qquad (15)$$

The algorithm completes after a given time has elapsed and then returns the obtained Pareto front approximation $\mathcal{P}$.

C. Genetic Algorithm

In the case of large problem instances, a general approach is the use of a Genetic Algorithm (GA) [20], which has been successfully applied to VRP [21] and other combinatorial optimization problems like scheduling [22]. Inspired by Darwin's theory of evolution, GA operates on an entire pool of solutions, called the population, instead of a single solution like TS.

The initial population is created in three steps. In step 1, the greedy method described earlier is performed, yielding a number of solutions (a Pareto set). In step 2, each solution in the initial population is created from a random solution from the greedy Pareto set. Since that introduces many repetitions (clones), in step 3 each solution is modified by performing $\frac{n}{10}$ random feasible swaps, increasing genetic diversity. Afterwards the algorithm proceeds iteratively. In each iteration genetic operators (selection, crossover and mutation) are applied, after which the Pareto front $\mathcal{P}$ must be updated.

Parent solutions for the crossover operator are chosen using the tournament method. The size of the tournament is square root of the population size, rounded to an integer. TOPSIS method is used to rank the solutions in the tournament and the best one wins. In the crossover process, each parent is randomly paired with another parent. Two crossover points are randomly selected in the first parent. The resulting fragment between the points is put as the starting portion of the child. If the fragment ended with 0 (vehicle end), we proceed immediately. Otherwise, we try to pack more orders into the current vehicle using the orders from the second parent (in order of appearance). Each time the current vehicle becomes infeasible, we close it and the order that created infeasibility is assigned to the next (empty) vehicle instead. Only orders that have not yet been used are added this way. This process repeats until all remaining orders are assigned to vehicles. The crossover probability is always 100%. Mutation is achieved by performing a single random swap move and the resulting solution is kept if it is feasible. The mutation probability is set to 15%. This is higher than usual in order to account for infeasible solutions.

After crossover and mutation are applied, the old population is replaced by children, except that the best 3% of the old population are kept regardless, a mechanism that is called elitism. Like TS, the stopping criterion is a time limit, and the Pareto front $\mathcal{P}$ is returned as a result.

V. Computer Experiment

In this section we present the results of the numerical experiment using instances based on real-life data as well as preliminary research for the greedy algorithm. The research methodology is described as well.

A. Instances, implementation and environment

In order to verify the effectiveness of the algorithms in a more realistic scenario, we have prepared a number of

Fig. 2. Distribution of parcel lockers locations against the map of the city of Wrocław

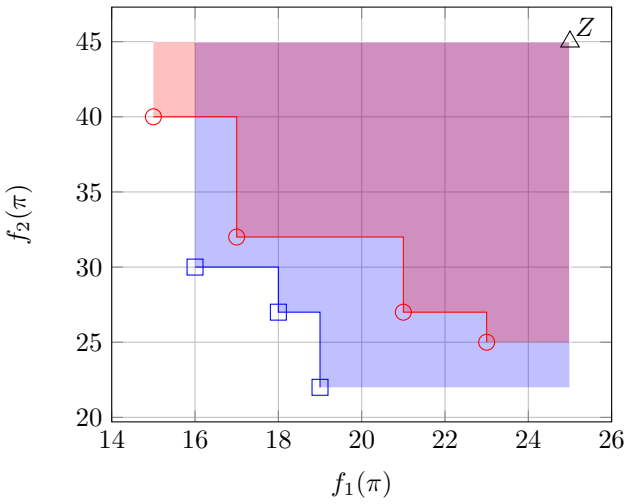

Fig. 3. Illustration of HVI with two fronts. Front indicated with squares has higher HVI

problem instances based on real-life data. To this end, we have collected locations of 344 parcel lockers and 3 possible depot locations used by a real-life delivery company operating in and around the city of Wrocław, Lower Silesia, Poland. Open Street Map [23] was then used to obtain realistic travel times $t_{k,l}$ between all location pairs. The city of Wrocław was used due to its geographical and infrastructure features that make the problem more challenging (many rivers and waterways, 111 bridges and many one-way streets). With this data we can generate problem instances with up to 344 parcel lockers.

For our experiment we generated the testing problem instances as follows. We consider 7 values for the number of orders $n \in \{500, 1000, 1500, \ldots, 3500\}$. For each n we consider three different numbers of vehicles $v \in \{0.1n, 0.3n, 0.5n\}$ and three different numbers of locations $l \in \{75, 150, 300\}$. This results in 63 instance groups. For each group we generate 10 instances, meaning 630 instances in total. By $x \sim \mathcal{U}\{a, b\}$ we denote that x follows a uniform integer distribution from a (inclusive) to b (inclusive). Then each instance is generated under the following further assumptions:

1) All l locations are randomly chosen from the original 344 possibilities (wihout repetitions).
2) Depot is chosen randomly from the 3 available options.
3) $a_i \sim \mathcal{U}\{300, 720\}$. In minutes since midnight. Indicates that orders are arriving between 5am and noon.
4) $d_i \sim \mathcal{U}\{a_i + 60, a_i + 180\}$. In minutes since midnight. Indicates that order deadline is 1 to 3 hours.
5) $g_i \sim \mathcal{U}\{1, l\}$.
6) $w_i \sim \mathcal{U}\{1, 20\}$. In kilograms.
7) $p_i \sim \mathcal{U}\{1, 10\}$.
8) $S = 2$ i.e. order service time is 2 minutes.
9) $P = 2$ i.e. vehicle time to park and leave is 2 minutes.
10) $C = 1000$. In kilograms (i.e. vehicle can take up to 1 metric ton of cargo).
11) $T = 480$. In minutes (i.e. vehicle time limit is 8 hours).

All algorithms were implemented in C# v. 12 programming language, compatible with .NET 8.0. All computer experiments were conducted on a DGX A100 machine with AMD EPYC 7742 3.2 GHz processor with 64 physical cores, 128 threads and 503 GiB of RAM working under Ubuntu 20.04.6 LTS operating system. The stopping condition for both TS and GA solving methods was set to $0.1n$ i.e. for $n = 1000$ the algorithm stops after 100 seconds. The implementation code is available at Github [24].

B. Measure of quality

There are several issues regarding how to compare the results of the proposed algorithms. Firstly, the algorithms return Pareto fronts (or rather the approximation of the real Pareto front), which can contain multiple solutions. To compare such results we use the concept of Hyper-Volume Indicator (HVI) [25]. For a two-criteria case and fronts $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$ the HVI of front $\mathcal{F}_i$ is calculated as the area of a figure constrained by points from $\mathcal{F}_i$ and a reference (or „nadir") point Z. The point Z is calculated by taking the worst values from all considered fronts and multiplying them by some constant (common value is 1.2, which is the same in our case). The concept of HVI is illustrated in Figure 3. Not that the higher HVI is, the better.

Secondly, the HVI provided by the tested algorithms should be compared to some baseline HVI. Such normalization will also remove the issue of different instances having vastly different HVIs. Thus, for a given instance I we calculate the Percentage Relative Deviation (PRD) as follows:

$$\mathrm{PRD}_{\mathrm{TS}}(I) = 100\% \frac{\mathrm{HVI}_{\mathrm{TS}}(I) - \mathrm{HVI}_{\mathrm{ref}}(I)}{\mathrm{HVI}_{\mathrm{ref}}(I)}, \qquad (16)$$

where $\mathrm{HVI}_{\mathrm{TS}}(I)$ is the HVI value achieved by TS algorithm for instance I and $\mathrm{HVI}_{\mathrm{ref}}(I)$ is HVI value for reference solution. Since considered problem sizes are too large to employ exact methods, we cannot use the actual (optimal) Pareto front. Instead, we use the HVI of the front obtained by the greedy method as our reference HVI. Thus, $\mathrm{PRD}_{\mathrm{TS}}(I) = 30\%$ will indicate that TS obtained 30% larger HVI for instance I. The value for the GA methods is calculated similarly:

$$\mathrm{PRD}_{\mathrm{GA}}(I) = 100\% \frac{\mathrm{HVI}_{\mathrm{GA}}(I) - \mathrm{HVI}_{\mathrm{ref}}(I)}{\mathrm{HVI}_{\mathrm{ref}}(I)}. \qquad (17)$$

TABLE II
BEST AND WORST SORTING STRATEGIES FOR THE GREEDY ALGORITHM

Average relative HVI	Sorting strategy formula
Best 5 strategies	
1.000	$(d_i - a_i)/p_i$
0.970	$1/(a_i d_i p_i (d_i - a_i))$
0.911	$d_i/(a_i(d_i - a_i))$
0.899	$(d_i p_i)/(a_i w_i (d_i - a_i))$
0.881	$(d_i(d_i - a_i))/a_i$
Worst 5 strategies	
0.509	$d_i w_i p_i$
0.500	d_i
0.494	$((d_i - a_i)p_i)/a_i$
0.410	$a_i d_i (d_i - a_i)$
0.410	$a_i p_i (d_i - a_i)$

C. Preliminary research

We have conducted a preliminary research to find out the best order sorting strategy for the greedy algorithm. We considered 5 order parameters that can be used as part of the sorting key: 1) a_i, 2) d_i, 3) w_i, 4) p_i and 5) $d_i - a_i$. In other words the sorting is done according to the following key:

$$(a_i)^{\alpha} \cdot (d_i)^{\beta} \cdot (w_i)^{\gamma} \cdot (p_i)^{\delta} \cdot (d_i - a_i)^{\epsilon}. \tag{18}$$

where $\alpha, \beta, \gamma, \delta, \epsilon \in \{1, -1, 0\}$. It means that each variable either goes into the numerator, goes into the denominator or is removed from the sorting key altogether, respectively. We have assumed that all sortings are in non-decreasing order, since the reverse is achieved simply by swapping the numerator with the denominator. All $3^5 = 243$ combinations were tested by comparing relative average HVI obtained by them on 1000 smaller instances (different than those presented in earlier subsection). The results for the best five and worst five combinations are shown in Table II

The results indicate that the best strategy is to sort orders according to the non-decreasing time remaining per priority ratio i.e.:

$$\frac{d_i - a_i}{p_i}. \tag{19}$$

Curiously, this strategy turned out to be the best in all tested cases. We can also see that this sorting strategy allows up to 2.5 times improvement over some other strategies and twice the improvement over sorting by deadlines alone.

D. Main research

The primary research was conducted on the 630 instances described earlier. The aggregated results for various instance groups (with regards to n, v and l) as well as for all instances overall, are shown in Table III. We will start with the analysis of the first three columns which show the specific scenario in question and the average PRD values of TS and GA for instances in the considered scenario, respectively.

We first observe that both TS and GA outperformed the greedy method on regular basis by 15.6% and 18.8% on average respectively. Thus, considering all tested instances, both algorithms achieved similar performance, with GA being slightly ahead. However, the situation becomes more complex where specific instance groups are considered. The results for various number of orders strongly suggest that the effectiveness of TS decreases as n increases and becomes low for $n > 2000$. For GA the effect seems to be opposite, though the relation is less obvious. Nonetheless, it should be noted that average GA improvement compared to the greedy method does not drop below 14%. It should be also noted that while the performance of TS drops with increase in n it remains higher than the performance of GA up to $n = 1000$. A similar observation can be made with regards to changing number of vehicles. As v increases, the performance of TS slightly drops and performance of GA significantly increases.

Curiously, with regards to the number of locations, the relationship between TS and GA reverses: TS slightly improves with increase in l while GA slightly drops in performance. This is not the only advantage of TS over GA. While GA seems to be more reliable on average, TS allows for better improvements as can be seen for cases with $n = 500$. Specifically, the highest $\mathrm{PRD}_{\mathrm{GA}}$ value over all instances was 2.363 (meaning HVI over twice higher than for the greedy method), while the highest $\mathrm{PRD}_{\mathrm{TS}}$ was 5.887, which is over twice as much. The maximal PRD values for each instance groups are shown in the 5th and 6th columns of Table III. Those results indicate that in almost all scenarios (except for higher n values) the maximal $\mathrm{PRD}_{\mathrm{TS}}$ is significantly higher than $\mathrm{PRD}_{\mathrm{GA}}$, even if such values are obtained rarely.

The above observations lead us to the conclusions that both of the proposed algorithms excel in different scenarios, most likely with accordance to the No Free Lunch Theorem. The TS lower performance for higher numbers of orders could be caused by its difficulty to overcome getting trapped in local optima effectively, while GA can do so easier due to its probabilistic nature. On the other hand, TS searches the entire neighborhood and is deterministic, leading to better results on smaller instances with smaller solution space.

While both algorithms can be used separately, the observed results indicate that TS and GA could work well in tandem, covering for each other's weaknesses. In order to research this, we have assumed a scenario where both algorithms are run in parallel and the best out of the two obtained PRD values is chosen and denoted $\mathrm{PRD}_{\mathrm{BEST}}$. The results of this experimented are shown in the 4th column of Table III. This allowed for a significant improvement in almost all cases except for $n = 500$ and $n = 3500$. The average improvement over standalone TS and GA is over 11% and 14%, respectively.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have considered a variant of the time- and capacity-constrained Vehicle Routing Problem with multi-criteria goal function, in order to model a modern delivery system based on parcel lockers. Two-criteria goal function was used to minimize the total travel time as well as the total penalty for late delivery of parcels. Real-life data taken from existing parcel lockers location in the city of Wrocław, Poland was used to create benchmark problem instances.

TABLE III
AVERAGE AND MAXIMAL PRD VALUES OF TS AND GA SOLVING METHODS AGGREGATED OVER VARIOUS INSTANCE GROUPS

Scenario	Average $\mathrm{PRD}_{\mathrm{TS}}$	Average $\mathrm{PRD}_{\mathrm{GA}}$	Average $\mathrm{PRD}_{\mathrm{BEST}}$	Maximal $\mathrm{PRD}_{\mathrm{TS}}$	Maximal $\mathrm{PRD}_{\mathrm{GA}}$
Over all groups					
	1.156	1.188	1.319	5.877	2.363
With regards to number of orders					
$n = 500$	1.632	1.159	1.696	5.877	1.905
$n = 1000$	1.223	1.141	1.324	3.126	1.972
$n = 1500$	1.086	1.209	1.270	2.141	2.259
$n = 2000$	1.106	1.242	1.338	3.025	2.363
$n = 2500$	1.026	1.202	1.222	1.788	2.095
$n = 3000$	1.017	1.158	1.172	1.418	1.960
$n = 3500$	1.003	1.209	1.209	1.019	1.787
With regards to number of vehicles					
$v = 0.05n$	1.182	1.104	1.263	5.877	1.645
$v = 0.1n$	1.165	1.214	1.349	3.714	1.905
$v = 0.2n$	1.122	1.248	1.345	4.094	2.363
With regards to number of locations					
$l = 75$	1.146	1.198	1.317	4.094	2.074
$l = 150$	1.149	1.188	1.309	3.714	2.363
$l = 300$	1.173	1.179	1.331	5.877	2.284

We have proposed two metaheuristic solving methods—a Tabu Search method and a Genetic Algorithm—which we compared with each other and a heuristic greedy method. We have also proposed an effective sorting strategy for the greedy method. The results indicated that the Genetic Algorithm was slightly better overall and when the number of orders and vehicles increased, while Tabu Search was better when the number of locations increased. We have also shown that both algorithms complement each other and provide significantly better results in almost all tested instances.

We consider several possible directions for future work. Firstly, the problem could be extended to allow for pickup of orders instead of just deliveries. Secondly, more real-life data from other cities could be collected to simulate other real-life settings for this problem. Finally, the existing algorithms could be further improved (i.e. adding a long-term memory to Tabu Search, using a Memetic Algorithm or employing parallel computing for both methods).

REFERENCES

[1] M. Mohammadi, G. Rahmanifar, M. Hajiaghaei-Keshteli, G. Fusco, C. Colombaroni, and A. Sherafat, "A dynamic approach for the multi-compartment vehicle routing problem in waste management," *Renewable and Sustainable Energy Reviews*, vol. 184, p. 113526, 2023. [Online]. Available: https://doi.org/10.1016/j.rser.2023.113526

[2] M. Eusébio, T. Pinto, and M. Vieira, "The consistent vehicle routing problem: An application to the pharmaceutical supply chain," in *Optimization, Learning Algorithms and Applications*, A. I. Pereira, A. Mendes, F. P. Fernandes, M. F. Pacheco, J. P. Coelho, and J. Lima, Eds. Cham: Springer Nature Switzerland, 2024, pp. 424–437.

[3] C. Gracia, B. Velázquez-Martí, and J. Estornell, "An application of the vehicle routing problem to biomass transportation," *Biosystems Engineering*, vol. 124, pp. 40–52, 2014. [Online]. Available: https://doi.org/10.1016/j.biosystemseng.2014.06.009

[4] E. Ayyıldız, M. C. Şahin, and A. Taşkın, "A multi depot multi product split delivery vehicle routing problem with time windows: A real cash in transit problem application in istanbul, turkey," *Journal of Transportation and Logistics*, vol. 7, no. 2, pp. 213–232, 2023. [Online]. Available: https://doi.org/10.26650/JTL.2022.1113726

[5] A. Sadeghi, F. Aros-Vera, H. Mosadegh, and R. YounesSinaki, "Social cost-vehicle routing problem and its application to the delivery of water in post-disaster humanitarian logistics," *Transportation Research Part E: Logistics and Transportation Review*, vol. 176, p. 103189, 2023. [Online]. Available: https://doi.org/10.1016/j.tre.2023.103189

[6] J. Wen and L. Yisong, "Vehicle routing optimization of urban distribution with self-pick-up lockers," *IEEE*, pp. 1–6, 2016. [Online]. Available: https://doi.org/10.1109/LISS.2016.7854384

[7] H. Abdullahi, L. Reyes-Rubiano, D. Ouelhadj, J. Faulin, and A. A. Juan, "Modelling and multi-criteria analysis of the sustainability dimensions for the green vehicle routing problem," *European Journal of Operational Research*, vol. 292, pp. 143–154, 2021. [Online]. Available: https://doi.org/10.1016/j.ejor.2020.10.028

[8] B. Gulmez, M. Emmerich, and Y. Fan, "Multi-objective optimization for green delivery routing problems with flexible time windows," *Applied Artificial Intelligence*, vol. 38, 2024. [Online]. Available: https://doi.org/10.1080/08839514.2024.2325302

[9] G. Srivastava, A. Singh, and R. Mallipeddi, "NSGA-II with objective-specific variation operators for multiobjective vehicle routing problem with time windows," *Expert Systems With Applications*, vol. 176, p. 114779, 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.114779

[10] J. Duan, Z. He, and G. G. Yen, "Robust multiobjective optimization for vehicle routing problem with time windows," *Transactions on Cybernetics*, vol. 52, pp. 8300–8314, 2022. [Online]. Available: https://doi.org/10.1109/tcyb.2021.3049635

[11] I. Orenstein, T. Raviv, and E. Sadan, "Flexible parcel delivery to automated parcel lockers: models, solution methods and analysis," *EURO Journal on Transportation and Logistics*, vol. 8, pp. 683–711, 2019. [Online]. Available: https://doi.org/10.1007/s13676-019-00144-7

[12] W. Li, L. Kunpeng, P. N. R. Kumar, and Q. Tian, "Simultaneous product and service delivery vehicle routing problem with time windows and order release dates," *Applied Mathematical Modelling*, vol. 89, pp. 669–687, 2021. [Online]. Available: https://doi.org/10.1016/j.apm.2020.07.045

[13] B. Pan, Z. Zhang, and A. Lim, "Multi-trip time-dependent vehicle routing problem with time windows," *European Journal of Operational Research*, vol. 291, pp. 218–231, 2021. [Online]. Available: https://doi.org/10.1016/j.ejor.2020.09.022

[14] T. Cokyasar, A. Subramanyam, and O. Sahin, "Time-constrained capacitated vehicle routing problem in urban e-commerce delivery," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2677, pp. 190–203, 2022. [Online]. Available: https://doi.org/10.1177/03611981221124592

[15] J. Grabenschweiger, K. F. Doerner, R. F. Hartl, and M. W. P. Savelsbergh, "The vehicle routing problem with heterogeneous locker boxes," *Central*

European Journal of Operations Research, vol. 29, pp. 113–142, 2021. [Online]. Available: https://doi.org/10.1007/s10100-020-00725-2

[16] R. Idzikowski, J. Rudy, and M. Jaroszczuk, "Solving a vehicle routing problem for a real-life parcel locker-based delivery," in *International Conference on Dependability of Computer Systems*. Springer, 2024, pp. 69–79.

[17] C.-L. Hwang and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications A State-of-the-Art Survey*. Springer Berlin, Heidelberg, 1981.

[18] R. Idzikowski, J. Rudy, and A. Gnatowski, "Solving non-permutation flow shop scheduling problem with time couplings," *Applied Sciences*, vol. 11, no. 10, 2021. [Online]. Available: https://doi.org/10.3390/app11104425

[19] J. L. Viegas, S. M. Vieira, E. M. P. Henriques, and J. M. C. Sousa, "A tabu search algorithm for the 3d bin packing problem in the steel industry," in *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control*, A. P. Moreira, A. Matos, and G. Veiga, Eds. Cham: Springer International Publishing, 2015, pp. 355–364.

[20] N. M. Razali, "An efficient genetic algorithm for large scale vehicle routing problem subject to precedence constraints," *Procedia-Social and Behavioral Sciences*, vol. 195, pp. 1922–1931, 2015.

[21] H. Park, D. Son, B. Koo, and B. Jeong, "Waiting strategy for the vehicle routing problem with simultaneous pickup and delivery using genetic algorithm," *Expert Systems with Applications*, vol. 165, p. 113959, 2021.

[22] X. Xia, H. Qiu, X. Xu, and Y. Zhang, "Multi-objective workflow scheduling based on genetic algorithm in cloud environment," *Information Sciences*, vol. 606, pp. 38–59, 2022.

[23] D. Luxen and C. Vetter, "Real-time routing with OpenStreetMap data," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '11. New York, NY, USA: ACM, 2011, pp. 513–516. [Online]. Available: https://doi.org/10.1145/2093973.2094062

[24] [hidden for review], "Github repository," 2024, accessed on 04.10.2024. [Online]. Available: [hidden-for-review]

[25] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999. [Online]. Available: https://doi.org/10.1109/4235.797969