Trading off costs and service rates in a first-mile ride-sharing service

Minyi Zheng, Giovanni Pantuso *

Department of Mathematical Sciences, University of Copenhagen, Copenhagen, 2100, Denmark

A R T I C L E   I N F O

Keywords:
Constrained multi-objective optimization problem
First-mile ride-sharing problem
Evolutionary algorithm

A B S T R A C T

Given a set of geographically dispersed vehicles, the first-mile ride-sharing problem seeks optimal routes to transport customers to a common destination (e.g., a transit station) via shared trips. In this article, we address the trade off between operating costs and service rates. The resulting multi-objective optimization problem is of a combinatorial nature. To obtain Pareto solutions we propose an evolutionary algorithm based on efficient non-dominated solution sorting. Extensive tests based on real-life data illustrate that the algorithm is able to find high-quality non-dominated fronts.

1. Introduction

We consider a first-mile ride-sharing service which employs a fleet of vehicles to transport people to a common destination (e.g., a transit station) by means of shared rides. Particularly, we study the problem of assigning transport requests to vehicles and determining their routes to the final destination. Transport requests are dispersed within the business area and are characterized by a latest arrival time at the destination. Vehicles are likewise dispersed as a result of ongoing activities and characterized by heterogeneous residual capacity. The problem is characterized by two conflicting objectives. On the one hand, the operator wishes to find a least cost set of feasible routes to transport customers to the end point. On the other hand, the operator wishes to maximize service rates, i.e., satisfy as many requests as possible. The problem is henceforth referred to as the multi-objective first-mile ride-sharing problem (MOFMRSP) and will be formally introduced in Section 3. Observe, thus, that the naming “ride-sharing” emphasizes the characteristic of a service where the operator allows multiple passengers to simultaneously occupy the same vehicle and is not to be confused with peer-to-peer sharing frameworks where users share their own vehicles.

As the size and population of urban areas increase, and with them road congestion and air pollution (Taniguchi et al., 2014) ride-sharing has been identified as a potential instrument to reduce the number of private cars on the road, emissions and congestion (Al-Abbasi et al., 2019). The potential of such services is still largely undiscovered. According to the New York City taxicab data (New York City Taxi and Limousine Commission, 2020), during January 2020 only 6% of the trips to Penn Station – a fairly popular transit station located in Manhattan – were shared. The remaining taxi trips to the station transported an individual passenger. This leaves significant room for coordinated on-demand transport.

The existing literature is likewise studying first-mile ride-sharing service from various points of view. Bian and Liu (2019b,a), Bian et al. (2020) and Chen and Wang (2018) focused on the design of pricing mechanisms. Shen et al. (2018) assessed the performance of a system that integrated a fleet autonomous vehicles for on-demand transportation, and public transportation. The authors focused on a case study in Singapore. Wang (2019) and Chen et al. (2020) built mathematical optimization models for addressing routing and order dispatching decisions. Wen et al. (2018) studied the impact of rebalancing mechanisms. Finally, Ye et al. (2022) studied fleet sizing decisions under different revenue-sharing mechanisms between the operator and the drivers. Nevertheless, the multi-objective nature of this problem, thus the trade-off between costs and levels of service, has not yet been explored. Such trade-off is crucial...
both at the design phase, e.g., when deciding the size of the fleet and of the business area, and at the operating phase, e.g., to allocate requests in order to preserve certain levels of service.

The literature presents a variety of methods for solving multi-objective optimization problems, both exact and heuristic, see, e.g., the surveys by Ehrgott and Gandibleux (2000) and Halfmann et al. (2022). Among these, Evolutionary Algorithms (EAs), a class of stochastic search algorithms that simulate the evolution of species, have attracted increasing attention. In general, EAs for multi-objective optimization problems (MOPs) can be divided into three main categories based, respectively, on indicators (García et al., 2021; Yuan et al., 2021; Liu et al., 2019), on decompositions (Jaszkiewicz, 2002; Zhang and Li, 2007; Liu et al., 2013) and on Pareto dominance (Deb et al., 2002; Kim et al., 2004; Corne et al., 2000). Algorithms based on indicators evaluate the convergence and diversity of the populations generated by means of metrics such as hyper-volume (Czyżak and Jaszkiewicz, 1998), inverted generational distance (Rostami and Neri, 2016; Zitzler and Thiele, 1998) and δ-binary quality indicator (Zitzler et al., 2003). These algorithms typically bear the burden of expensive repeated computation the indicators that guide the search. Algorithms based on decomposition decompose a MOP into several single-objective problems or into a set of simpler MOPs. This type of algorithms has recently received significant attention, see, e.g., Jaszkiewicz (2002), Zhang and Li (2007) and Liu et al. (2013). They employ a set of weights e.g., uniformly distributed, that represent the importance of each individual objective. Each setting of the weight generates a separate subproblem, and guides the algorithm to converge in the direction set by the weights. This typically guarantees a sufficient diversity in the population. However, this type of algorithms might become ineffective for highly constrained problems as the weights may fail to generate a sufficiently diverse population. Finally, algorithms based on Pareto dominance select solutions (e.g., for reproduction or survival) using non-dominated sorting operators (Deb et al., 2002; Kim et al., 2004; Corne et al., 2000). Such algorithms are typically cheaper from a computational perspective.

In this article we develop an EA based on Parato dominance for the MOFMRSP. Particularly, we use the sorting procedure referred to as Non-dominated Sorting Genetic Algorithm II (NSGA-II) proposed by Deb et al. (2002) which ensures an $O(mN/sup/2)$ computational complexity, where $m$ is the number of objectives and $N$ is the population size. The contributions of this article can be stated as follows.

1. We introduce a mathematical model for the MOFMRSP. The mathematical model extends the available literature by simultaneously addressing, for the first time, the cost-service rate trade-off under detailed online operating constraints, e.g., arrival times of existing and new passengers and residual vehicles capacity.

2. We propose a constrained multi-objective evolutionary algorithm based on Pareto dominance with a single-parent generation operator for finding high quality non-dominated solutions to the MOFMRSP. The algorithm extends the available literature as it introduces new, problem-specific, encoding as well as crossover and mutation operators.

3. We provide empirical evidences on the performance of the algorithm based on extensive numerical experiments on real-life data. Particularly, the availability of extensive taxi data, such as that provided by New York City Taxi and Limousine Commission (2020), which contains detailed trip records for several years beginning with 2009, allows validating the algorithm in a realistic environment. This is consistent with other studies on ride-sharing services which use the same data set, see e.g., Al-Abbasi et al. (2019).

The remainder of this article is organized as follows. In Section 2 we briefly recall some of the central concepts related to constrained multi-objective optimization. In Section 3 we formally introduce the MOFMRSP and express it as a mathematical problem. In Section 4, we introduce the proposed EA and in Section 5 we demonstrate its performance on extensive numerical experiments. Finally, we draw conclusions in Section 6.

2. Constrained multi-objective optimization

A constrained multi-objective optimization problem (CMOP) is defined as

$$\min \ F(x) := (f_1(x), f_2(x), \ldots, f_m(x))^T$$  \hspace{1cm} (1a)

subject to

$$g_i(x) \geq 0 \quad i = 1, \ldots, q$$  \hspace{1cm} (1b)

$$x \in \Omega$$  \hspace{1cm} (1c)

where $x \in \mathbb{R}^n$ is a vector of decision variables, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \ldots, m$ define $m$ conflicting objectives, and inequalities $g_i(x) \geq 0$, with $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \ldots, q$, define the solution space. Finally, $\Omega \subseteq \mathbb{R}^n$ may restrict the value of $x$ to e.g., integer values. In this article we are particularly concerned with instances of CMOPs where all $f_i$ and $g_i$ are linear in $x$. We proceed now by providing some basic definitions which will be useful in the remained of this treatise.

**Definition 2.1 (Feasible Solution).** A solution $x$ is feasible if $g_i(x) \geq 0$ for all $i = 1, \ldots, q$.

**Definition 2.2 (Feasible Region).** The feasible region $S \subseteq \Omega$ is the set of all feasible solutions.

The definition of optimality in CMOPs is slightly more involved and requires the concept of dominance. Let the degree of constraint violation of a solution $x$ at the $i$th constraint, $i = 1, \ldots, q$, be defined as

$$G_i(x) := \min\{0, g_i(x)\}$$  \hspace{1cm} (2)
and the total degree of constraint violation of solution $x$ be defined as

$$ G(x) := -\sum_{i=1}^{n} G_i(x). $$

This entails that $x$ is feasible if $G(x) = 0$ and infeasible if $G(x) > 0$. We can now introduce the concept of constrained dominance.

**Definition 2.3 (Constrained Dominance).** Let $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$ be two solutions to CMOP (1). Then $x$ is said to dominate $y$, indicated by $x \prec y$, if any of the following conditions holds

1. $G(x) = 0$ and $G(y) > 0$
2. $0 < G(x) < G(y)$
3. $0 = G(x) = G(y)$ and $f_i(x) \leq f_i(y)$ for all $i = 1, 2, ..., m$, and there exists at least one $j \in \{1, 2, ..., m\}$ such that $f_j(x) < f_j(y)$.

That is, the first condition states that a feasible solution always dominates an infeasible solution. The second condition states that, if $x$ and $y$ are both infeasible, then the solution with the smallest degree of constraint violation dominates the other. Finally, the third condition states that $x$ dominates $y$, if $x$ and $y$ are both feasible but $x$ has objective values not worse that those of $y$ for all objectives, and strictly better than $y$ for at least one objective.

**Definition 2.4 (Pareto Optimal).** A solution $x$ is said to be a Pareto optimal or a Pareto solution if there is no solution that dominates $x$.

**Definition 2.5 (Pareto Front).** Let the set of Pareto optimal solutions be defined as

$$ PS = \{ x_1, x_2, ..., x_k \} $$

We defines the Pareto front as the set of outcomes of the Pareto optimal solutions in $PS$, that is

$$ PF = \{ F(x_1), F(x_2), ..., F(x_k) \} $$

In remainder of this article, references to the sets $PF$ and $PS$ are used somewhat interchangeably. That is, we might refer, with an abuse of terminology, to a solution in the Pareto front while solutions are in $PS$ and only their outcomes are in $PF$. In these cases, we always mean a solution whose outcome is in the Pareto front.

Fig. 1 provides a qualitative description of the Pareto front for a bi-objective problem. The filled circles represent the outcomes $PF$ of the Pareto optimal solution in $PS$. It can be noted that no solution dominates the solutions in the Pareto front. The solutions represented by empty circles are instead dominated by at least one other solution.

3. First-mile ride-sharing problems

We consider a set of vehicles $K$ employed in a first-mile ride-sharing service. The vehicles have common capacity $Q$, and each vehicle $k \in K$ is available from time $T_k$ at location $o(k)$ having $Q_k$ passengers on board (thus residual capacity $Q - Q_k$), as a result of ongoing activities. This, in practice, makes the fleet heterogeneous in the problem. A set of new transport requests $P$ is collected, where each request $i \in P$ is characterized by a requested arrival time $A_i$ and a pick-up position $o(i)$. All customers have a common destination (e.g., a transit station) denoted $d$ and located at position $o(d)$. Similarly, the passengers on-board the vehicle at the
beginning of the planning horizon determine a latest arrival time $A_k$ for each vehicle $k \in \mathcal{K}$ (if the vehicle is initially empty one can set $A_k = T^U$, where $T^U$ is a generally valid arrival time upper bound). All travel times and costs are assumed to be known in advance. Particularly, we let $C_{ij}$ and $T_{ij}$ be the travel cost and travel time, respectively, between locations $o(i)$ and $o(j)$ with $i \in \mathcal{K} \cup \mathcal{P}$ and $j \in \mathcal{P} \cup \{d\}$.

The decision maker is to assign transport requests to vehicles such that the capacity of the vehicles and the requested arrival times are respected. Two conflicting objectives guide the decision. On the one hand, the decision maker wishes to find a minimum-cost set of routes to transport customers. On the other hand, they wish to maximize the service rate, defined as the number of requests fulfilled over the total number of requests received. We refer to this problem as the Multi-Objective First-Mile Ride-Sharing Problem (MOFMRSP).

An example instance with nine requests and six vehicles is depicted in Fig. 2. We observe that the station is (arbitrarily) located in the center of the business area and customers (circles) and vehicles (rectangles) are spread out around the station. We can also observe a potential solution which satisfies a number of requests. As an example vehicle $k_3$ picks up customers $p_2$ and $p_3$ and, as requested, terminates the journey at the station. Three requests, namely $p_7$, $p_8$ and $p_9$, will not be satisfied and two of the vehicles, namely $k_2$ and $k_6$, will remain at their original location.

In order to express the MOFMRSP in mathematical terms, we let decision variables $x_{kij}$ take value 1 if vehicle $k$ drives directly from location $o(i)$ to location $o(j)$ with $i \in \mathcal{K} \cup \mathcal{P}$ and $j \in \mathcal{P} \cup \{d\}$ and $x = (x_{kij})_{i \in \mathcal{K} \cup \mathcal{P}, j \in \mathcal{P} \cup \{d\}, k \in \mathcal{K}}$. Furthermore, we let decision variables $q_i$, $i \in \mathcal{P} \cup \{k\}$ represent the number of passengers on board a vehicles upon picking up customer $i$. The MOFMRSP can be formulated as follows.

$$
\min_x f_1(x) := \sum_{i \in \mathcal{K} \cup \mathcal{P}, j \in \mathcal{P} \cup \{d\}, k \in \mathcal{K}} C_{ij}x_{kij}
$$

$$
\max_x f_2(x) := \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}, j \in \mathcal{P} \cup \{d\}, k \in \mathcal{K}} x_{kij}
$$

s.t. \quad \sum_{j \in \mathcal{P} \cup \{d\}} x_{kji} \leq 1 \quad \forall k \in \mathcal{K}

$$
\sum_{j \in \mathcal{P} \cup \{d\}} x_{kji} = \sum_{i \in \mathcal{K} \cup \mathcal{P}} x_{kij} \quad \forall k \in \mathcal{K}
$$

$$
\sum_{i \in \mathcal{K} \cup \mathcal{P}} x_{kij} = \sum_{i \in \mathcal{L}} x_{kji} \quad \forall j \in \mathcal{P}, \forall k \in \mathcal{K}
$$
elimination constraints. Observe that the vehicle, that a vehicle can arrive at a customer location, vehicle must then depart from the customer, and constraints (4g) ensure that a customer is picked up at most once. Notice, in (4e) its original location, its journey terminates at the station. Constraints (4e) state that whenever a vehicle picks up a customer, the maximizng the service rate expressed as the number of customers picked up over the total number of customer requests. Constraints of constraints (4j) reduces to the smallest between \(A\), \(\forall i \in P \cup K\) and constraints (4f) or (5) . Constraints (4h) ensure that the capacity of the vehicles is respected. Constraints (4i) state that, if \(i \in K \cup P\), \(j \in P \cup \{d\}, k \in K\) otherwise it reduces to \(A\) in any case. Finally, constraints (4k) set the \(x_{ij}^k\) as binary variables, constraints (4l) set the bound on the \(q_i\) variables, and constraints (4m) ensure that there are no connections between a location and itself.

4. Multi-objective evolutionary algorithm

This section presents a multi-objective evolutionary algorithm (EA) for solving the MOFMRPSP. In a nutshell, EAs imitate the evolution of species and work, at each iteration (a generation), with a population of individuals. Each individual represents a solution to the problem. Appropriate measures of fitness are used to select the individuals that reproduce and survive to the next generation. Reproduction happens by means of appropriately defined mathematical operators (Eiben et al., 2003) that generate new individual from existing ones. In the remainder of this section, we first provide an high level overview of the algorithm in Section 4.1 and then explain its components in detail in Section 4.2 through Section 4.6.

4.1. Overview

Algorithm 1 provides a high level overview of the EA developed. The algorithm requires the definition of a stopping criterion and a single parameter, \(N\) indicating the population size. It starts (line 3) by randomly generating an initial population \(T^0\) with \(N\) individuals. Subsequently (line 4), the EliteSelection operator selects all the non-dominated feasible solutions in the initial population. These solutions will form the initial version of the elite set \(A^0\). Thus the algorithm enters the main loop which terminates only when the stopping criterion is met. We assume, as in the experiments presented in Section 5, that the algorithm terminates after a fixed number of generations \(G\). In the main loop, the current population \(T^{g-1}\) is evolved (line 7). The Evolution operator takes as input the current population and returns a set \(O^g\) of offspring. Then, at line 8, the SurvivalSelection operator chooses \(N\) individuals from the union of current population and offspring to form the next population \(T^g\). Note that is allowed to carry infeasible solutions to the next generation. Following (line 9), the elite set \(A^g\) is updated by choosing the feasible non-redundant non-dominated solutions from the current population \(T^g\). Observe thus that \(A^g \subseteq T^g\). Finally, at line 11 the algorithm returns the elite set after the stopping criterion has been met.
satisfaction of arrival times is not guaranteed. To avoid the possibility that a vehicle might remain idle for a long time, we increase the chances of generating feasible solutions with respect to arrival times. Nevertheless, the solutions generated may violate arrival time constraints (4j).

4.3. Initialization

(4i) which forbid non-empty vehicles to remain idle. Nevertheless, the solutions generated may violate arrival time constraints (4j).

Finally, it should be noted that the this encoding automatically respects routing and capacity constraints as well as constraints (4i) which forbid non-empty vehicles to remain idle. Nevertheless, the solutions generated may violate arrival time constraints (4j).

4.2. Encoding

We encode a solution (individual in the EA jargon) using a matrix \( V \in \mathbb{R}^{K \times Q} \), where each row describes a vehicle and each column a seat in the vehicle. The entries in the matrix are customers or a zero when a seat remains empty. Seats are occupied in the order in which customers are visited. For instance, matrix (6) describes an example individual where it is assumed that the problem counts four vehicles (the matrix has four rows) and five customers, and that the capacity \( Q \) of the vehicles is four (the matrix has four columns). Negative entries in the matrix represent customers that are already on board the vehicle. In this case \( Q_3 = 2 \) as vehicle 3 has already two passengers on board. A row with all zero entries indicates a vehicle that remains empty and idle in its original position. In this example, no passengers are assigned to vehicle 4 which will therefore remain idle. Note that it is by chance that all customers in the example are included in the matrix. The encoding allows for picking up only a subset of the customers. A customer which is not an entry in the matrix is not picked up.

Finally, it should be noted that the this encoding automatically respects routing and capacity constraints as well as constraints (4i) which forbid non-empty vehicles to remain idle. Nevertheless, the solutions generated may violate arrival time constraints (4j).

4.3. Initialization

The initial population is generated using a semi-random procedure which has the twofold goal of covering as large a search area as possible as well as to prevent extensive generation of infeasible solutions with respect to arrival times.

The generation of the initial population proceeds as follows:

1. The business area is partitioned into four regions as shown in Fig. 3.
2. Vehicles and customers are assigned to the region where they are located. As an example, in Fig. 3 vehicle 1 is assigned to Region 3, vehicle 2 is assigned to Region 2, vehicle 3 to Region 1 and vehicle 4 to Region 4. Customers \( p_1, p_2 \) and \( p_3 \) are assigned to Region 3, while customers \( p_1 \) and \( p_4 \) are assigned to Region 2. Let \( P_i \) and \( K_i \) be the set of customers and vehicles, respectively, assigned to region \( i = 1, \ldots, 4 \).
3. For each region \( i = 1, \ldots, 4 \):
   
   (a) \( P \leq |P_i| \) customers from \( P_i \) are selected at random with equal probability.
   (b) Each customer selected is randomly assigned, with equal probability, to a vehicles in \( K_i \) with empty seats available.

Observe that the procedure described ensures that the capacity of the vehicles is respected. In addition, by assigning customers to vehicles in the same region, we increase the chances of generating feasible solutions with respect to arrival times. Nevertheless, satisfaction of arrival times is not guaranteed.

\[
\begin{pmatrix}
\text{Seat}_1 & \text{Seat}_2 & \text{Seat}_3 & \text{Seat}_4 \\
2 & 1 & 5 & 0 \\
4 & 3 & 0 & 0 \\
-1 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{pmatrix}
\]
4.4. Evolution

At each iteration of the algorithm the population evolves by generating offspring. This is done in such a manner that maintains and combines desirable features of the parents. Particularly, we adopt unary crossover and mutation operators which generate one child solution from each individual in the current population. The procedure is described in Algorithm 2.

Consider, as an example, the individual shown in Eq. (6) and assume it is the parent from which we generate a new individual. The crossover operator first randomly chooses an integer \( J \leq |K|/2 = 2 \). Assume the chosen number is 1. Then chooses two different vehicles, assume \( k_1 \) and \( k_2 \), and assigns them into the only group, \( G_1 \). Assume the routes are depicted in Fig. 4(a) (for the sake of space the position of customers and vehicles is different than in the example show in Fig. 3). Then, the algorithm randomly swaps customer 2 from the first vehicle and customer 3 from the second vehicle. The result is shown in matrix (7) and illustrated in Fig. 4(b), where it can be noticed that the route has margins for improvements. Assume no mutation is performed at line (16). Algorithm 2 rearranges the customers in the route at line 20 such that every vehicle travels the shortest distance to the station. The resulting individual is provided in (8) and its graphical representation in Fig. 4(c).

\[
\begin{bmatrix}
\text{Seat}_1 & \text{Seat}_2 & \text{Seat}_3 & \text{Seat}_4 \\
3 & 1 & 5 & 0 \\
4 & 2 & 0 & 0 \\
-1 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]  
(7)

\[
\begin{bmatrix}
\text{Seat}_1 & \text{Seat}_2 & \text{Seat}_3 & \text{Seat}_4 \\
5 & 1 & 3 & 0 \\
4 & 2 & 0 & 0 \\
-1 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]  
(8)
Algorithm 2 Crossover and Mutation operator

1: **Input** (i) \( T \): Current population (ii) \( \mu \): Mutation probability
2: \( \mathcal{O} = \emptyset \)
3: for Individual \( i \) in \( P \) do
4: Randomly choose an integer \( J \leq |K|/2 \)
5: Randomly choose \( 2J \) different vehicles
6: Randomly divide the selected vehicles into \( J \) groups of two
7: Denote the groups as \( G_1, G_2, \ldots, G_J \)
8: for Vehicle group \( G_j \), \( j = 1, \ldots, J \) do
9: if Both vehicles in \( G_j \) do not include current customers then
10: Do nothing
11: else
12: Randomly select a seat on of the two vehicles and swap their content
13: end if
14: Let \( R \) be a random sample from \( U(0,1) \)
15: if \( R < \mu \) then
16: Select at random a non-selected customer and assign it to a randomly chosen vehicle from \( G_j \), if it has residual capacity
17: end if
18: end for
19: for Vehicle \( k \) in \( K \) do
20: Rearrange the order of the passengers on board in order to minimize the travel distance
21: end for
22: Assign the new individual to \( O \)
23: end for
24: **Output** \( \mathcal{O} \): offspring

Observe that, at line 12 of Algorithm 2, different cases may materialize depending on whether the seats selected from the vehicles are occupied or not by passengers. The example provided represents the case in which both seats selected are occupied by passengers. In this case, the passengers simply change vehicle. Another case which may materialize at line 12 is that one of the two vehicles has seats occupied by previous customers (e.g., seats one and two in vehicle 3). In this case, the random seat selection is limited to the seats which are not occupied by previous customers. In the example individual (6), the random seat selection on vehicle 3 would be limited to seats 3 and 4. A final case which may materialize is that the seat selected for a swap is empty. Consider for example the case in which seat four is selected from vehicle 2 and seat three is selected from vehicle 1. In this case, customer 5 moves from vehicle 1 to vehicle 2 and no customer from vehicle 2 to vehicle 1. Note that, when customer 5 is placed on seat four of vehicle 2, seat three remains empty. However, at line 20 the order of the passengers will be rearranged and passengers 5 will occupy seat one, two or three, depending on the order in which it is picked up in the optimized route. Finally, observe that selecting seats rather than passengers allow the possibility of moving an individual customer from a vehicle to another without performing a swap and thus to explore different solutions.

Finally, note that the route optimization of line 20 does not take into account passengers already on board, but simply the route from \( \omega(k) \) to the station, via the new customers assigned.

4.5. Evaluation

After offspring has been generated, all the available individuals are evaluated in order to allow the selection of the elite set and of the new population.

The following procedure is adopted to evaluate individuals. For each individual in \( T \cup \mathcal{O} \)

1. Check whether the individual is feasible with respect to arrival time constraints.
2. If the individual is feasible compute \( f_1 \) and \( f_2 \).
3. If the individual is infeasible repair it (see the procedure below) and then you compute \( f_1 \) and \( f_2 \).

Observe that, consistently with (4a) and (4b), the computation of cost and service rate does not take into account customers that are already on board the vehicle when planning. As an example, when computing the cost and service rate of (6), the two passengers on board vehicle 3 are not computed in the service rate, and the cost of its route only accounts for the traveling between the original position of vehicle 3 and the station. In fact, the cost for picking up the customers already on board vehicle 3 has been accounted for in a previous planning phase or are already born. Finally, note that feasibility concerns only arrival time constraints (4j).

A simple repair procedure is adopted which attempts, but does not guarantee, to provide a feasible solution from an infeasible one. Given an infeasible individual, the repair procedure finds the vehicles which violate the arrival time constraints (4j). From
Fig. 4. Graphical representation of the crossover and mutation operator applied on the individual in Eq. (6).

(a) Graphical representation of the routes of vehicles $k_1$ and $k_2$ in the parent solution.

(b) Graphical representation of the new individual in (7).

(c) Graphical representation of the offspring in (8).
these vehicles, it excludes the passenger with the earliest arrival time. Observe that passengers already on board, such as passengers
−1 and −2 in vehicle 3 of solution (6), cannot be excluded. Note, however, that meaningful instances always guarantee that the
travel time between 𝑥(𝑘) and the station is such that the desired arrival time 𝐴𝑘 of the passengers already on-board can be satisfied.

4.6. Elite and survival selection

After each individual in the current population and offspring has been evaluated, the algorithm proceeds to select those who
will be part of the next generation and of the elite set. The selection procedure is sketched in Algorithm 3.

Algorithm 3 Survival and elite selection

1: Input 𝑇folk ∪ 𝐿folk: Population and offspring at generation 𝑔
2: Input 𝑁: population size
3: Output 𝑇folk: Next generation, 𝑀folk: Elite set after generation 𝑔
4: Divide 𝑇folk−1 ∪ 𝐿folk into non-dominated sets 𝑆1, 𝑆2, …, 𝑆𝑛
5: 𝑀folk = {𝐼 ∈ 𝑆𝑖| 𝐼 is feasible }
6: 𝑇folk = ∅
7: while |𝑇folk| < 𝑁 do
8: Let 𝐼 = min { 𝑗| 𝑆𝑖 ≠ ∅ }
9: Let 𝐼 = arg max { 𝑖∈𝑆𝑖} 𝐶𝐷(𝐼) 
10: Set 𝑆 organisation = 𝑆organisation \ { 𝐼 }
11: Set 𝑇folk = 𝑇folk ∪ { 𝐼 }
12: end while

The algorithm starts by partitioning 𝑇folk−1 ∪ 𝐿folk into non-dominated sets 𝑆 organisation 1, 𝑆 organisation 2, …, 𝑆 organisation 𝑛 , where 𝑆 organisation 1 contains the non-dominated solutions from 𝑇folk−1 ∪ 𝐿folk, 𝑆 organisation 2 the non-dominated solutions in (𝑇folk−1 ∪ 𝐿folk)\ 𝑆 organisation 1 and 𝑆 organisation 𝑗 the non-dominated solutions in (𝑇folk−1 ∪ 𝐿folk)\ 𝑈organisation=1 𝑆 organisation 𝑖 . Fig. 5 qualitatively illustrates the outcomes of three non-dominated sets. Such partition is obtained using the NSGA-II sorting criterion by Deb et al. (2002). Note that the partitioning is performed using solely the objective values for the dominance criteria. That is, we do not adopt the dominance criteria for constrained optimization described in Section 2. Thus, in this case an infeasible solution may dominate a feasible one, depending solely on the values of the respective objective functions. This allows us to carry, from a population to another, high-potential infeasible solutions.

At line 5 the elite set is populated by selecting from 𝑆 organisation 1 (thus from the non-dominated solutions in the entire population) the individuals which are feasible with respect to the arrival times.

Following, at line 7 the algorithm starts selecting solutions to keep in the next population 𝑇folk. It selects individuals picking, in increasing order, from the top non-dominated sets. That is, it begins choosing from 𝑆 organisation 1 , and once all its individuals have been added to 𝑇folk, it selects individuals in 𝑆 organisation 2 , than in 𝑆 organisation 1 and so on, until 𝑁 elements have been chosen. Particularly, among the individuals in a given set 𝑆 organisation 𝑗 , it selects in non-increasing order of the crowding distance 𝐶𝐷(𝐼), a measure of closeness between solutions. The highest the crowding distance, the further away a solution is from its neighbors. By selecting individuals with high crowding distance one
improves diversification in the population. Particularly, the crowding distance for an individual $I$ is computed as

$$CD(I_i) = \sum_{j=1}^{m} |f_j(I_{i+1}) - f_j(I_{i-1})|$$

where $m$ is the number of objective functions, $I_{i+1}$ and $I_{i-1}$ are the individuals whose outcomes are closest to the outcome of $I_i$. Observe that the crowding distance of the individuals whose outcomes are closest to the axes is set to $\infty$, and that, in case of ties, a random individual is chosen. The concept of crowding distance is graphically depicted in Fig. 6. At the end of the procedure, the algorithm has chosen $N$ individuals following a non-dominance criteria, and among them those with the highest measure of diversity.

5. Experimental study

In this section we report on the effectiveness of the proposed algorithm based on empirical evidences from an experimental study. The algorithm was implemented in Matlab and all tests were performed on a computer with dual-core 64-bit 2.7 GHz, equipped with Intel Core i5 CPU and 8 GB of RAM.

Particularly, we designed a two-phase experiment. In the training phase we tune the parameters of the algorithm on a set of small instances. In the test phase we use the tuned algorithm on a set of larger instances to collect empirical evidences on its performance.

This section is organized as follows. We start by presenting the instances used in the training and test phases in Section 5.1. Following, in Section 5.2 we introduce two indicators that will help us assessing the results of the algorithm. In Section 5.3 we report the results of the training phase and in Section 5.4 we report the results of the test phase. Finally, in Section 5.5 we illustrate a potential use of the algorithm in the context of fleet planning.

5.1. Instances

Our benchmark instances are constructed using actual taxi trip records occurred in the city of New York sourced from (New York City Taxi and Limousine Commission, 2020). We assume that the focal station is located at Times Sq/Theatre District, see Fig. 7 location 230, due to its central position in the Manhattan area. We use as a reference data-set all the taxi trips to zone 230 performed between January 1st and January 31st 2021, between 7:00 am and 7:30 am.

Customers pick-up locations and vehicle locations are generated by randomly drawing actual taxi trip records from the reference data-set and using the pick-up location in the trip record as the customer’s or vehicle’s location. However, taxi trip records only contain pick-up and drop-off zones, not actual coordinates. Therefore, we generate customers and vehicles location by perturbing the coordinates of the zone. The perturbation is however small enough to ensure that the location remains within the zone. Customer’s arrival times are generated by adding 10 min to the arrival time in the trip record drawn. All distances are then computed as Euclidean distances. For all instances we assume unit fuel cost $C = 1$, unit velocity $V = 40$ km/h, capacity $Q = 4$ and all vehicles empty at startup ($Q_k = 0$ for all $k \in K$).

For the training phase we generate in total 15 instances of three different sizes. The number of customers ranges from $|P| = 5$ to $|P| = 15$, while the number of vehicles ranges from $|K| = 2$ to $|K| = 4$ in order to generate a total capacity (number of seats)
slightly exceeding the number of customers. The training instances are listed in Table 1 where the class identifies the size of the instance and for each class there are five randomly generated instances.

For the test phase we generate two sets of test instances namely a set of small test instances, listed in Table 2, and a set of large test instances, listed in Table 3. The former is used to compare the solutions of the algorithm against the true Pareto front. The latter is used to assess the performance of the algorithm on larger instances for which the true Pareto front is not necessarily known.

Table 1
Training instances.

<table>
<thead>
<tr>
<th>Class</th>
<th>Instance</th>
<th>Customers</th>
<th>Vehicles</th>
<th>Total capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>C₅ V₂ 1</td>
<td>5</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>C₅ V₂ 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₅ V₂ 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₅ V₂ 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₅ V₂ 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C₂</td>
<td>C₁₀ V₃ 1</td>
<td>10</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>C₁₀ V₃ 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₁₀ V₃ 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₁₀ V₃ 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₁₀ V₃ 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C₃</td>
<td>C₁₅ V₄ 1</td>
<td>15</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>C₁₅ V₄ 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₁₅ V₄ 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₁₅ V₄ 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₁₅ V₄ 5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2
Small test instances.

<table>
<thead>
<tr>
<th>Class</th>
<th>Instance</th>
<th>Customers</th>
<th>Vehicles</th>
<th>Total capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₄</td>
<td>C₈ V₂ 1</td>
<td>8</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>C₈ V₂ 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₈ V₂ 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C₅</td>
<td>C₁₀ V₃ 1</td>
<td>10</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>C₁₀ V₃ 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C₁₀ V₃ 3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 7. TLC taxi zones in Manhattan.
The set of small test instances includes six instances with eight to ten customers and two to three vehicles in such a way to obtain a total capacity equal to or slightly exceeding the number of customers. The instances are listed in Table 2 where we can notice that each class contains three randomly generated instances. The set of large test instances includes 15 instances where the number of customers ranges from $|\mathcal{C}| = 20$ to $|\mathcal{C}| = 40$ and the number of vehicles from $|\mathcal{V}| = 5$ to $|\mathcal{V}| = 10$. Again, the size of the fleet is chosen in order to have a total capacity equal to or slightly above the number of customers. The instances are listed in Table 3, where we can notice that each class contains five randomly generated instances of equal size.

5.2. Indicators

Obtaining the true Pareto front might be challenging in practical applications. In these cases, one often works with an estimated Pareto front. Usually, the estimated Pareto front has to meet two requirements (Zitzler et al., 2004), namely convergence and diversity. Convergence entails that the estimated Pareto front is “close” to the true Pareto front. Diversity entails that the Pareto front includes a variety of outcomes. Appropriate indicators measure the degree to which each of these criteria is fulfilled. Over the past decade, this issue has been widely studied, see e.g., Czyzak and Jaszkiewicz (1998), Zitzler and Thiele (1998), Zitzler et al. (2000), Deb and Jain (2002), Zitzler et al. (2003) and Knowles et al. (2006). Two indicators are particularly suitable for the problem at hand. These are the inverted generational distance (IGD) indicator proposed by Czyzak and Jaszkiewicz (1998) and the hypervolume (HV) indicator proposed by Zitzler and Thiele (1998).

The IGD indicator calculates the average distance between the outcomes in the real Pareto front and the nearest solution found in the generated Pareto front. Let $P$ be the generated Pareto front and $PF$ the true Pareto front, see Definition 2.5. The IGD is computed as follows

$$\text{IGD}(P, PF) = \frac{\sum_{o^* \in PF} \left( \min_{o \in P} \{d(o^*, o)\} \right)}{|PF|}$$

(9)
where $d_{o_1o_2}$ is the Euclidean distance between point $o_1$ and $o_2$. That is, the IGD measures the average shortest distance between the points in $P^*$ and the points in $P$. Fig. 8 depicts the elements involved in the calculation of the IGD indicator for a fictitious bi-objective problem.

The HV indicator is used to calculate the (high-dimensional) total volume enclosed by the obtained front and a given reference point whose coordinates represent (for minimization problems) an upper bound on the $m$ conflicting objectives. Large HV values signal both convergence and diversity as the generated front is, intuitively, far from the reference point, thus close to the true front, and wide, thus representing a diverse set of combinations of the objective values. In our experiments we use the HV to compare the (smaller) volume generated by the front delivered by the algorithm to the (larger) volume generated by the true or reference front. The closer is the volume of the delivered front to that of the true front, the higher is the quality of the front obtained.

For a given front the HV is calculated as follows

$$HV = \delta \left( \bigcup_{o=1}^{|P|} v_o \right),$$

where $\delta$ is the Lebesgue measure, $P$ represents the set of points in the generated front, and $v_o$ represents the hypercube formed by the reference point and the $o$th point in the generated front. The HV is qualitatively depicted in Fig. 9 for a fictitious bi-objective problem. It can be noticed that, in the two-dimensional space, the Lebesgue measure represents the area of the union of the rectangles formed by the points in the front and the reference point $z$.

5.3. Results of the training phase

A key parameter in evolutionary algorithms is the total number of individuals the algorithm evaluates during the entire duration of its operations. We refer to this number as objective functions evaluations (OFEs). The number of OFEs is given by the product between the number of generations $G$ and the population size $N$, i.e., $OFEs = G \times N$. This entails that, there are two central decisions to be made, that may influence the performance of the algorithm, namely (i) the total number of OFEs, and (ii) how the OFEs are to be distributed across generations, i.e., values for $G$ and $N$, given OFEs. Our training phase is set to answer these questions.

We start by finding appropriate parameters to express $OFEs$ as a function of the size of the instance and, once a suitable value for $OFEs$ for a given instance size is available, we determine suitable values for $G$ and $N$, given OFEs. Our training phase is set to answer these questions.

We use the hypervolume (HV) as a metric to assess the quality of the solutions obtained. The HV metric simultaneously measures diversity and convergence of the Pareto front generated. The calculation of the HV requires a reference point whose coordinates are bounds on the objective values. Particularly, we use point $z = (1.1, \bar{C})$ where 1.1 is an upper bound on the service rate, and $\bar{C}$ is an upper bound on the cost computed as (assuming the triangular inequality holds)

$$\bar{C} = \sum_{i \in K} \max_{j \in P} \left\{ C_{i,j} \right\} + (Q - 1) \max_{i \in P, j \in P} \left\{ C_{i,j} \right\} + \max_{i \in P} \left\{ C_i, d \right\}$$

$z$: Reference point
$o_i$: Outcome in the generated front
HV

**Fig. 9.** Graphical representation of the HV indicator.
assumes that each vehicle travels the highest possible distance. That is, it sums the highest distance between the origin of $C_i$ is marginal compared to the configuration with $\alpha$ includes random choices—see Section 4). Additional statistics can be found in Appendix. The value of $\alpha$ is only marginal with $\alpha$ appears to give the highest average HV value for most of the instances. The improvement achieved with $\alpha = 200$, thus with more solutions evaluated, is marginal compared to the configuration with $\alpha = 100$. Based on these evidences, we use the value of $\alpha = 100$ for the test phase.

Table 5 reports, for different $\alpha$-values, the average HV values across five runs of the algorithm (observe that the algorithm includes random choices—see Section 4). Additional statistics can be found in Appendix. The value of $\alpha = 100$ appears to give the highest average HV value for most of the instances. The improvement achieved with $\alpha = 200$, thus with more solutions evaluated, is marginal compared to the configuration with $\alpha = 100$. Based on these evidences, we use the value of $\alpha = 100$ for the test phase.

Further evidences can be obtained by assessing the number of generations it took the algorithm to converge for different values of $\alpha$. Fig. 10 illustrates how the HV evolves along with the generations for different values of $\alpha$ on three training instances, namely $C5.V2.1$, $C10.V3.1$ and $C15.V4.1$. Particularly, it can be noticed that the algorithm plateaus earlier with $\alpha = 100$, an the speed up is only marginal with $\alpha = 200$. Based on these evidences, we use the value of $\alpha = 100$ for the test phase.

Table 7 reports the average HV metric over five runs of the algorithm for different values of $\beta$ (additional statistics are provided in Appendix). It can be observed that the value of $\beta = 1 : 4$ yields most frequently the highest average HV values. Fig. 11 illustrates the HV progression with the number of generations on three example instances. It can be noted that the algorithm plateaus faster with $\beta = 1 : 10$ and $\beta = 1 : 4$ compared to $\beta = 10 : 1$. This indicates that with $\beta = 10 : 1$ the population size is not large enough.

Table 4

| $\alpha$ | Class | $|P|$ | $OFEs$ | $N$ | $G$ |
|---|---|---|---|---|---|
| 60 | $C_1$ | 5 | 300 | 15 |
|  & $C_2$ | 10 | 600 | 30 | 20 |
|  & $C_3$ | 15 | 900 | 45 |
| 100 | $C_1$ | 5 | 500 | 25 |
|  & $C_2$ | 10 | 1000 | 50 | 20 |
|  & $C_3$ | 15 | 1500 | 75 |
| 200 | $C_1$ | 5 | 1000 | 50 |
|  & $C_2$ | 10 | 2000 | 100 | 20 |
|  & $C_3$ | 15 | 3000 | 150 |

Table 5

<table>
<thead>
<tr>
<th>Instance</th>
<th>$a = 60$</th>
<th>$a = 100$</th>
<th>$a = 200$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C5.V2.1$</td>
<td>30.8281</td>
<td>30.9064</td>
<td><strong>30.9165</strong></td>
</tr>
<tr>
<td>$C5.V2.2$</td>
<td>79.1403</td>
<td>79.1415</td>
<td>79.0092</td>
</tr>
<tr>
<td>$C5.V2.3$</td>
<td>76.9345</td>
<td>76.9713</td>
<td>76.8872</td>
</tr>
<tr>
<td>$C5.V2.4$</td>
<td>33.5072</td>
<td>33.8296</td>
<td><strong>33.8537</strong></td>
</tr>
<tr>
<td>$C5.V2.5$</td>
<td>47.8173</td>
<td>47.8288</td>
<td><strong>47.8996</strong></td>
</tr>
<tr>
<td>$C10.V3.1$</td>
<td>143.9875</td>
<td>144.0186</td>
<td>143.8547</td>
</tr>
<tr>
<td>$C10.V3.2$</td>
<td>160.1736</td>
<td>160.4745</td>
<td>159.8795</td>
</tr>
<tr>
<td>$C10.V3.3$</td>
<td>128.9652</td>
<td>129.0445</td>
<td>128.9770</td>
</tr>
<tr>
<td>$C10.V3.4$</td>
<td>135.5856</td>
<td>136.8786</td>
<td>136.7188</td>
</tr>
<tr>
<td>$C10.V3.5$</td>
<td>334.9141</td>
<td>334.9446</td>
<td>334.9587</td>
</tr>
<tr>
<td>$C15.V4.2$</td>
<td>243.3641</td>
<td>242.9838</td>
<td>242.9312</td>
</tr>
<tr>
<td>$C15.V4.3$</td>
<td>202.1637</td>
<td>202.7148</td>
<td>201.7020</td>
</tr>
<tr>
<td>$C15.V4.5$</td>
<td>483.3735</td>
<td>483.2706</td>
<td><strong>483.4776</strong></td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Class</th>
<th>$OFEs$</th>
<th>$G$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 : 10$</td>
<td>$C_1$</td>
<td>500</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>$C_2$</td>
<td>1000</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>$C_3$</td>
<td>1500</td>
<td>15</td>
<td>100</td>
</tr>
<tr>
<td>$1 : 4$</td>
<td>$C_1$</td>
<td>500</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>$C_2$</td>
<td>1000</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>$C_3$</td>
<td>1500</td>
<td>20</td>
<td>75</td>
</tr>
<tr>
<td>$10 : 1$</td>
<td>$C_1$</td>
<td>500</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$C_2$</td>
<td>1000</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>$C_3$</td>
<td>1500</td>
<td>100</td>
<td>15</td>
</tr>
</tbody>
</table>
and the algorithm runs for many iterations before converging, with a higher risk of falling into local optima. With $\beta = 1 : 10$, on the other hand, the algorithm is not able to complete as many generations as with $\beta = 1 : 4$. Bases on these evidences we select the ratio $\beta = 1 : 4$ to be used in the test phase.
Summarizing, in the Test Phase we use the values provided in Table 8 for the parameters of the algorithm.

5.4. Results of the test phase

In the test phase we use our tuned algorithm to solve the test instances. The scope of the test phase is to obtain evidences on the quality of the Pareto fronts delivered by the algorithm. Particularly, we use a set of small test instances, listed in Table 2, to compare the Pareto fronts delivered by the algorithm to the true Pareto fronts. In addition, we use a set of larger test instances, listed in Table 3, to obtain evidences on the Pareto fronts obtained for larger problems for which obtaining the true Pareto front is impractical.

The true Pareto front to the small test instances is obtained by solving the following single-objective problem for different values of $\gamma$.

$$\min_{x} f_1(x) := \sum_{i \in K, p \in P \cup \{d\}} \sum_{j \in K} \sum_{k \in K} C_{ij} x_{ij}$$  \hspace{1cm} (11a)

s.t. (4d)–(4m)

$$\frac{1}{|P|} \sum_{i \in P} \sum_{j \in P \cup \{d\}} \sum_{k \in K} x_{ij} = \gamma$$  \hspace{1cm} (11b)

Problem (11) minimizes travel costs while ensuring a fixed service rate through constraints (11b). Particularly, we solve problem (11) for $\gamma \in \left\{ \frac{|P|}{|P|}, \frac{|P|-1}{|P|}, \ldots, \frac{|P|}{2|P|} \right\}$, that is we focus on the portion of the Pareto front with solutions that ensure that at least half of the customers are picked up. Since the image of $f_2$ is a finite countable set and comprises the above mentioned $\gamma$ values, this
allows us to obtain the true Pareto front, provided that we hold only the solution with the highest value of $\gamma$ whenever we find more solutions with the same value of $f_1$ for different $\gamma$ values.

Fig. 12 plots the true Pareto fronts of the small instances together with the Pareto fronts delivered by ten different runs of our algorithm. In each run the algorithm is allowed to run for at most 10 min or as many OFEs as provided in Table 8, whichever comes first. It can be noticed that the generated Pareto fronts are fairly close to, and often even coincide with, the true Pareto fronts. This suggests that the algorithm is able to deliver high quality Pareto front in a rather consistent manner, despite some variation inherent in the heuristic nature of the method proposed.

The quality of the Pareto fronts obtained by the algorithm is also certified by the statistics on the IGD indicator reported in Table 9. It can be noted that the IGD indicator is consistently close to zero, which indicates that the true and the generated Pareto fronts are rather close to each other.

To assess the performance of the algorithm on the large test instances we rely on the HV and IGD indicators. However, since obtaining the true Pareto front for these instances is impractical, we use as benchmark a simulated Pareto front obtained by solving each test instance 30 times using our tuned algorithm, and recording all the non-repeated non-dominated solutions obtained. We consider this set of solution outcomes our reference or simulated Pareto front.

Fig. 13 depicts the generated Pareto fronts delivered by ten runs of the algorithm (also in this case in each run the algorithm is allowed to run for at most 10 min or as many OFEs as provided in Table 8, whichever comes first) compared with the simulated Pareto fronts. Visually, it appears that the generated Pareto fronts are fairly close to the simulated Pareto fronts and, at times, partially overlapping. This suggests that the algorithm is rather consistent, despite some variation between the different runs.

To assess the quality of the Pareto fronts numerically, we evaluate the IGD and HV values. Table 10 provides the worst, average and best IGD value over ten runs of the algorithm for all the 15 test instances. We recall that the IGD indicator is used to measure the closeness of the solutions to the true (simulated in this case) Pareto front, thus the smaller the IGD value, the better the performance of the algorithm, with a zero IGD value indicating that the algorithm has found the true Pareto front. We observe that the IGD is on average close to 1, thus slightly higher than for the small test instances. However, its value does not seem to increase dramatically with the size of the instances, e.g., between the $C_{20}V_{5}$, * and the $C_{40}V_{10}$, * instances despite the latter instances are twice as large. This indicates that the algorithm scales sufficiently well.

Table 11 presents the average HV value as a percentage of the HV value of the simulated Pareto front. Observe that the HV represents a volume, thus we are comparing the average volume covered by the solutions obtained to the volume covered by the simulated Pareto front. We observe that, in all cases, the Pareto fronts obtained cover more than 99% of the volume covered by the simulated Pareto front. This indicates that the quality and diversity of the solutions of our algorithm are very close to that of the simulated Pareto front, and thus that the algorithm converges to high quality and sufficiently diversified Pareto fronts.
Fig. 13. Pareto fronts obtained during ten runs of the algorithm compared to the simulated Pareto front.
Table 10
Worst, average and best IGD values over ten runs of the algorithm on the large test instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Worst</th>
<th>Mean</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>C20_V5_1</td>
<td>1.2057</td>
<td>0.8723</td>
<td>0.5595</td>
</tr>
<tr>
<td>C20_V5_2</td>
<td>1.4199</td>
<td>0.9974</td>
<td>0.6112</td>
</tr>
<tr>
<td>C20_V5_3</td>
<td>1.9998</td>
<td>1.1761</td>
<td>0.5866</td>
</tr>
<tr>
<td>C20_V5_4</td>
<td>2.4813</td>
<td>1.3612</td>
<td>0.5871</td>
</tr>
<tr>
<td>C20_V8_1</td>
<td>0.9525</td>
<td>0.6729</td>
<td>0.5015</td>
</tr>
<tr>
<td>C20_V8_2</td>
<td>2.5268</td>
<td>1.4199</td>
<td>0.3419</td>
</tr>
<tr>
<td>C20_V8_3</td>
<td>3.7095</td>
<td>1.7418</td>
<td>0.5713</td>
</tr>
<tr>
<td>C20_V8_4</td>
<td>3.9205</td>
<td>1.6681</td>
<td>0.5869</td>
</tr>
<tr>
<td>C20_V8_5</td>
<td>2.4813</td>
<td>1.3612</td>
<td>0.5871</td>
</tr>
<tr>
<td>C30_V10_1</td>
<td>1.2807</td>
<td>0.9307</td>
<td>0.6723</td>
</tr>
<tr>
<td>C30_V10_2</td>
<td>1.2518</td>
<td>0.9175</td>
<td>0.5460</td>
</tr>
<tr>
<td>C30_V10_3</td>
<td>1.3461</td>
<td>0.8777</td>
<td>0.4855</td>
</tr>
<tr>
<td>C30_V10_4</td>
<td>1.9986</td>
<td>1.1130</td>
<td>0.3568</td>
</tr>
<tr>
<td>C30_V10_5</td>
<td>2.1675</td>
<td>1.0269</td>
<td>0.4608</td>
</tr>
<tr>
<td>C40_V10_1</td>
<td>2.1300</td>
<td>1.1628</td>
<td>0.5756</td>
</tr>
</tbody>
</table>

Table 11
Worst, average and best HV indicator values over ten runs of the algorithm for all large test instances. HV values are expressed as a percentage of the HV value for the simulated Pareto front.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Worst</th>
<th>Mean</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>C20_V5_1</td>
<td>99.08%</td>
<td>99.27%</td>
<td>99.99%</td>
</tr>
<tr>
<td>C20_V5_2</td>
<td>99.29%</td>
<td>99.50%</td>
<td>99.63%</td>
</tr>
<tr>
<td>C20_V5_3</td>
<td>99.17%</td>
<td>99.34%</td>
<td>99.81%</td>
</tr>
<tr>
<td>C20_V5_4</td>
<td>99.01%</td>
<td>99.09%</td>
<td>99.46%</td>
</tr>
<tr>
<td>C20_V5_5</td>
<td>99.02%</td>
<td>99.13%</td>
<td>99.19%</td>
</tr>
<tr>
<td>C30_V8_1</td>
<td>99.18%</td>
<td>99.56%</td>
<td>99.94%</td>
</tr>
<tr>
<td>C30_V8_2</td>
<td>99.00%</td>
<td>99.43%</td>
<td>99.70%</td>
</tr>
<tr>
<td>C30_V8_3</td>
<td>99.11%</td>
<td>99.26%</td>
<td>99.74%</td>
</tr>
<tr>
<td>C30_V8_4</td>
<td>99.43%</td>
<td>99.66%</td>
<td>99.85%</td>
</tr>
<tr>
<td>C30_V8_5</td>
<td>99.47%</td>
<td>99.79%</td>
<td>99.97%</td>
</tr>
<tr>
<td>C40_V10_1</td>
<td>99.14%</td>
<td>99.78%</td>
<td>99.81%</td>
</tr>
<tr>
<td>C40_V10_2</td>
<td>99.19%</td>
<td>99.47%</td>
<td>99.82%</td>
</tr>
<tr>
<td>C40_V10_3</td>
<td>99.04%</td>
<td>99.30%</td>
<td>99.82%</td>
</tr>
<tr>
<td>C40_V10_4</td>
<td>99.42%</td>
<td>99.66%</td>
<td>99.82%</td>
</tr>
<tr>
<td>C40_V10_5</td>
<td>99.67%</td>
<td>99.77%</td>
<td>99.91%</td>
</tr>
</tbody>
</table>

Table 12
Instances used for fleet sizing demonstration and corresponding parameters of the algorithm.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>)</td>
<td>( OFEs )</td>
<td>( N )</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

5.5. Usage scenarios

In this section we demonstrate the usage of the algorithm for fleet sizing purposes. We assume the decision maker is to decide the size of the fleet taking both routing costs and service rates into account. Particularly, give a number of customers, we assess how different fleet sizes have an impact on routing costs for the same target service rate. That is, we assume that all investment costs and salary costs are sunk costs when routing decisions are made (e.g., drivers are not hired on a short-term basis).

We use the instances described in Table 12 which provide, for a given number of customers, different alternative fleet sizes. Fig. 14 reports the Pareto fronts generated for the instances in Table 12. It can be observed that the Pareto front is shifted below whenever the fleet size increases. This means that, given a target service level, a larger fleet can achieve that service level with lower operating costs. For example, in the instance with 20 customers (see Fig. 14(a)), a 100% service level would cost 40 with a fleet of five vehicles and less than 35 with a fleet of 8 vehicles. The difference in terms of operating costs can, in some cases, be remarkable, as for the highest service rates in Fig. 14(b).
Fig. 14. Service rate and cost over the instances with different amount of vehicles.

(a) Service rate and cost for the instances with 20 customers.

(b) Service rate and cost for the instances with 30 customers.

(c) Service rate and cost for the instances with 40 customers.
In this appendix we provide additional statistics on the performance of the algorithm in the training phase (see Tables 13 and 14).

6. Conclusion

This article formulated the problem of trading off operating costs and service rates in a first-mile ride-sharing service as a multi-objective mixed-integer programming problem. Its combinatorial nature makes the problem challenging to address. For this reason, the article introduces a specialized evolutionary algorithm. The algorithm devised ad-hoc reproduction and mutation operators based on the specific structure of the problem.

Extensive experiments on instances generated from real-life data indicate that the algorithm is able to deliver high quality Pareto fronts. Particularly, the algorithm is able to cover over 99% of a reference Pareto front generated, for benchmark purposes, using extensive runs of the algorithm.

CRediT authorship contribution statement

Minyi Zheng: Methodology, Investigation, Software, Data curation, Writing – original draft, Visualization. Giovanni Pantuso: Conceptualization, Methodology, Supervision, Writing – review & editing.

Appendix. Additional results