Multiobjective evolutionary algorithms for the taxi sharing problem

Renzo Massobrio*, Gabriel Fagúndez and Sergio Nesmachnow

Universidad de la República,
Julio Herrera y Reissig 565,
Montevideo, Uruguay
E-mail: renzom@fing.edu.uy
E-mail: gabrielf@fing.edu.uy
E-mail: sergion@fing.edu.uy
*Corresponding author

Abstract: Transportation planning plays a central role in the design and development of smart cities. In particular, the concept of sharing economy applied to urban transportation is gaining massive public attention in recent years. This article presents the application of two multiobjective evolutionary algorithms to the problem of distributing passengers travelling from the same origin to different destinations in several taxis. A new problem formulation is presented, accounting for two quality of service metrics from the point of view of taxi users: the total cost of the trips and the delay experienced by each passenger. Two multiobjective evolutionary algorithms are proposed: a parallel microevolutionary algorithm (following a linear aggregation approach to combine the problem objectives) and one well-known algorithm from the literature (following a full multiobjective approach based on Pareto dominance). Both algorithms are compared against each other and against two greedy heuristics based on the ideas presented in the related literature. The experimental evaluation is performed over a set of 88 problem instances generated using real GPS taxi data. Results show that the proposed algorithms are able to efficiently reach significant improvements in both problem objectives over the greedy heuristics in short execution times.

Keywords: evolutionary algorithms; multiobjective optimisation; taxi sharing.

Reference to this paper should be made as follows: Massobrio, R., Fagúndez, G. and Nesmachnow, S. (201x) 'Multiobjective evolutionary algorithms for the taxi sharing problem', *Int. J. Metaheuristics*, Vol. XX, No. YY, pp.XXX–XXX.

Biographical notes: Renzo Massobrio, Assistant at Universidad de la República, Uruguay, since 2015. Engineer in Computer Science from Universidad de la República, MSc candidate in Computer Science at Universidad de la Repúblicaand PhD candidate in Computer Science at Universidad de Cádiz, Spain. Renzo Massobrio received Best undergraduate thesis award from the Faculty of Engineering, Universidad de la República. Research internships at Universidad de Cádiz in Spain, Universidad de Málaga in Spain, Cardiff University in Wales and Centro de Investigación Científica y de Educación Superior de Ensenada in México. Main research interests are computational intelligence, metaheuristics and high-performance computing applied to solving complex optimisation problems and have published more than 10 papers on the subject.

Gabriel Fagúndez, Engineer in Computer Science from Universidad de la República, since 2015, received Best undergraduate thesis award from the Faculty of Engineering, Universidad de la República. Main research interests are computational intelligence and metaheuristics applied to solving urban transportation problems and have published 5 papers on the subject.

Sergio Nesmachnow, Full Professor at Universidad de la República, Uruguay, since 1997. Engineer, MSc and PhD in Computer Science from Universidad de la República, Uruguay. Post-doctorate studies at University of Luxembourg, University of Málaga and Centro de Investigación Científica y de Educación Superior de Ensenada in Baja California, México. Director of Postgraduate Studies, Engineering Faculty, Universidad de la República, Uruguay. Researcher at National Agency for Research and Innovation and National Program for the Development of Basic Sciences. Main research interests are scientific and high-performance computing, computational intelligence and metaheuristics and have published more than 45 journal papers, more than 150 conference papers, two books and seven book chapters on these topics. Since 2012, Editor in Chief of the International Journal of Metaheuristics (Inderscience Publishers) and Guest Editor of Cluster Computing (Springer) and The Computer Journal (Oxford Journals).

This paper is a revised and expanded version of a paper entitled ‘Multiobjective taxi sharing optimization using the NSGA-II evolutionary algorithm’ presented at *11th Metaheuristic International Conference*, Agadir, Morocco, 2015.

1 Introduction

The concept of *smart city* proposes the use of information and communications technologies with the goal of improving the quality and performance of urban services. In this way, it is possible to reduce costs, increase efficiency in the use of resources, and allow a more active participation of citizens (Deakin and Al Waer, 2011). Such techniques are often applied to transportation services due to the central role they play in modern cities.

In this context, *carpooling* has gained massive public attention in recent years (Fellows and Pitfield, 2000). Sharing vehicles among people with similar travel needs has both economical and environmental benefits, at individual and collective levels. This practice allows minimising the travel costs for passengers while simultaneously reducing the number of vehicles on the streets and pollution. Having less vehicles on the streets leads to fewer traffic jams, resulting in a more fluent, thus more efficient traffic. This contributes to minimising the impact of transportation in the environment, which is a major concern nowadays, specially for big cities (Ferrari et al., 2003; Katzev, 2003). The aforementioned benefits of car sharing have led to initiatives to attend the public concern on this topic. Exclusive carpool lanes, campaigns to promote car sharing, and a plethora of mobile applications to find carpooling mates are some of the examples that illustrate the importance of this subject (Pfanner, 2012).

Taxis are a fast and reliable mean of transportation. However, high fares often discourage potential users. Consequently, taxis rarely run at full capacity. Therefore, taxis could benefit from the same principles that apply to carpooling. There are some web platforms that provide solutions for ride-sharing scheduling both for taxis and private vehicles. However, most applications simply act as a billboard where users can search for travel partners or, in the best case, use very simple algorithms to suggest companions based on their proximity.

Some research works have studied different variants of the taxi-sharing problem. Since the taxi-sharing problem is NP-hard (Letchford, Eglese and Lysgaard, 2002), heuristics and metaheuristics are needed to find high-quality solutions in reasonable execution times when dealing with realistic problem instances. This article presents two multiobjective evolutionary algorithms (MOEAs) to solve the multiobjective one-origin-to-multiple-destinations variant of the taxi-sharing problem. The experimental evaluation over real-world scenarios demonstrates that the proposed methods are accurate and efficient tools to solve the problem and can be easily integrated in online (web, mobile) applications.

This article extends our previous conference paper, presented at the 11th Metaheuristics International Conference (Massobrio, Fagúndez, and Nesmachnow, 2015b) . Two MOEAs to solve the taxi-sharing problem are presented: one that follows a domain decomposition strategy and one with an explicit multiobjective approach. The experimental analysis compares the performance of both algorithms against greedy strategies based on the ideas presented in the related literature. A new set of problem instances is used to compare the performance of the different algorithms. The main contributions of the research reported in this article are the design and implementation of two MOEAs to solve the one-to-many taxi-sharing problem.

The article is organised as follows. Section 2 introduces the multiobjective version of the taxi-sharing problem. Section 3 reviews related work about taxi-sharing optimisation and related problems. Section 4 introduces evolutionary algorithms (EAs) and their multiobjective variants and describes the proposed MOEAs to solve the problem. Section 5 describes the problem instances used and reports the experimental results of the two algorithms proposed against two greedy heuristics to solve the problem. Finally, Section 6 formulates the conclusion and the main lines for future work.

2 Problem definition: multiobjective taxi sharing

This section introduces the taxi-sharing problem and its mathematical formulation as a multiobjective optimisation problem.

2.1 Problem description and model

The taxi-sharing problem models a reality where a number of people located in the same origin, decide to travel to different destinations using taxis. The problem proposes finding an appropriate number of taxis and the distribution of passengers, with the goal of simultaneously minimising two important objectives for taxi users: the cost of the trip and the perceived delay to reach their destinations. Our problem model considers traffic flow scenarios without major traffic jams. Including traffic data in real time is proposed as one of the main lines of current and future work.

Regarding the taxi-sharing problem, the following should be considered:

- Each taxi can transport a limited number of passengers depending on its capacity. The amount of taxis available for each capacity is given as input.
- The maximum number of taxis that can be used in a scenario with N passengers is N, in the case where each passenger travels in a separate vehicle.
- The cost for each taxi includes the cost to hire the taxi (minimum fare or flag-drop charge) and the cost for travelling from the origin to the final destination. No other costs are considered such as waiting times, tips, or extra baggage fees.

- Each passenger has an associated tolerance indicating the additional time they are willing to spend due to taxi sharing, compared to the time it would require a direct trip from the origin to their destinations. The tolerance of each passenger is also given as an input to the algorithm.

2.2 Mathematical formulation

This subsection introduces the mathematical formulation for the multiobjective taxi-sharing problem aimed at simultaneously minimising the cost of the travel and the delay perceived by the users to reach their destinations. Given the following elements:

- A set of passengers $P = \{p_1, p_2, \ldots, p_N\}$ starting from a common origin O and having a set of potentially different destinations $D = \{d_1, d_2, \ldots, d_N\}$. $dest : P \rightarrow D$ returns the destination of each passenger.
- A set of taxis $T = \{t_1, t_2, \ldots, t_M\}$, $M \leq N$, with capacities (passengers that each taxi can carry) $K = \{K_1, K_2, \ldots, K_M\}$.
- A function $C : T \rightarrow \mathbb{N}_0$ that determines how many passengers use a taxi on a given trip, where $C(t_i) \leq K_i$.
- A constant B that indicates the cost of hiring a taxi (flag drop charge).
- A distance function, $dist : \{\{O\} \cup D\} \times D \rightarrow \mathbb{R}_0^+$.
- A cost function associated to the distance travelled by each taxi, $cost : \mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$.
- A travel time function, $time : \{\{O\} \cup D\} \times D \rightarrow \mathbb{R}_0^+$.
- A tolerance function $tol : P \rightarrow \mathbb{R}_0^+$ stating the additional time that each passenger is willing to tolerate over the time of a direct trip from the origin O to their destination.

The problem aims to find a planning function to transport the N passengers in L taxis ($L \leq N$), i.e., a function $f : P \rightarrow T \times \mathbb{N}$, where $h \leq C(t_i)$, $\forall (t_i, h)/f(p_j) = (t_i, h)$ holds. Function f assigns passengers to taxis and determines the order in which the taxi will visit their destinations, simultaneously minimising the total cost given by function TC [Eq. (1)] and the total delay given by the function TD [Eq. (2)], where $f^{-1}(t_i, j)$ returns the passenger assigned to position j (order of delivery) in taxi t_i and $dest\big(f^{-1}(t_i, 0)\big) = O,\ \forall\, t_i$.

$$TC = \sum_{t_i, C(t_i) \neq 0} \left[B + \sum_{j=1}^{C(t_i)} cost\Big(dist\, \overbrace{\Big(dest\big(f^{-1}(t_i, j-1)\big), dest\big(f^{-1}(t_i, j)\big) \Big)}^{\text{consecutive destinations in the route of taxi } t_i} \Big) \right] \quad (1)$$

$$TD = \sum_{t_i} \left[\sum_{j=1}^{C(t_i)} \left[\overbrace{\sum_{h=1}^{j} time\Big(dest\big(f^{-1}(t_i, h-1)\big), dest\big(f^{-1}(t_i, h)\big) \Big)}^{\text{effective riding time of passenger in position } j \text{ of taxi } t_i} - \left[\underbrace{tol\big(f^{-1}(t_i, j)\big) + time\Big(O, dest\big(f^{-1}(t_i, j)\big)\Big)}_{\text{time tolerated by passenger in position } j \text{ of taxi } t_i} \right] \right] \right] \quad (2)$$

3 Related work

This section reviews the related works on the literature about taxi-sharing and related optimisation problems.

The taxi-sharing problem is a variant of the *carpooling problem (CPP)*. Hartman et al. (2014) studied the CPP as a graph theory problem, considering two different scenarios and presenting greedy heuristics for each of them:

i a scenario where the set of drivers is known in advance

ii a scenario where it is unknown who will act as drivers and who as passengers.

Although the first scenario can be solved exactly in polynomial time (Hopcroft and Karp, 1971), a greedy heuristic is proposed to deal with large instances in reasonable execution times. The greedy algorithm finds solutions closer than 4% to the optimum on average. Two greedy heuristics are presented for the second scenario, which is $\mathcal{NP}$-hard (Knapen et al., 2015). The experimental analysis is performed using traffic simulations from the region of Flanders, Belgium. Both algorithms are compared in terms of quality of the solutions and computational efficiency. However, the amount and size of problem instances is not reported and no comparisons against other algorithms from the related literature are performed.

A problem closely related to the CPP is the *dial-a-ride problem (DARP)*. The main difference with the CPP is that vehicle owners are not in the set of passengers willing to ride-share. Cordeau and Laporte (2003) solved the static *DARP* for a set of vehicles that start from a common depot and have to fulfill trip request from a group of passengers. The goal is to find the routes that minimise the operational cost of the fleet of vehicles. Each passenger has a time window for pickup and drop. An upper bound for the time one passenger can stay in a vehicle is added as a hard restriction to the problem. The problem is modelled using a graph and is solved using a Tabu search heuristic. To prevent the search from quickly converging to local optima, a relaxation mechanism is proposed, which allows the search to explore unfeasible solutions. The experimental analysis is performed over a set of 20 instances (having between 24 and 144 requests each) randomly generated using realistic information from Montreal, Canada for time windows, vehicle capacity, and trip's duration. Results show that the algorithm is able to find solutions less than 1.5% far from the optimum in execution times varying between 2 and 90 minutes depending on the instance.

Santos and Xavier (2013, 2015) studied the dynamic *DARP* with money as an incentive. The problem consists in assigning trip requests to taxi drivers in order to maximise the amount of served requests and minimise the total cost incurred by passengers. The mathematical formulation considers the different capacities of the vehicles and the time window of each passenger request. The two conflicting objectives are combined into a single-objective function using an aggregation approach. The problem is proven to be $\mathcal{NP}$-hard by applying an analogy to the metric hamiltonian path problem. The proposed solution consists in discretising the day into short periods of time and solving a static version of the problem for each period. A greedy randomised adaptive search procedure (GRASP) is proposed to solve each static instance. The evaluation is performed over a set of 24 problem instances, generated using two different travel patterns from North-eastern Illinois, USA of 698 and 744 requests each, varying the time windows for passengers and the number of taxis available. Additionally, four problem instances were manually

generated for the city of Sao Paulo, Brazil. In both cases, the travel costs between destinations is indirectly computed by calculating the distance and assuming a constant vehicle speed. The distance used is that of a straight line in the latitude/longitude space, not considering the real route that a vehicle needs to follow to connect two points in the map. An average cost improvement of 18.25% is reported comparing to a scenario where trips are not shared. Later (Santos and Xavier, 2015), the authors incorporated a path-relinking algorithm to the GRASP heuristic. Path relinking takes two solutions to the problem and applies specific operations iteratively, in order to make the first solution more similar to the second one. In each iteration, a new solution is found, which may be better than the two solutions first considered. The path-relinking heuristic allows outperforming the previous GRASP algorithm, computing solutions with average cost savings of 30%.

Manna and Prestwich (2014) studied the online stochastic ride-sharing problem with time windows. The formulation consists in maximising the number of shared rides while minimising the delay experienced by each passenger. Time windows are considered as soft constraints, allowing reasonable delays expected in realistic scenarios. The objective function considers both the current passenger-to-vehicle assignment as well as all possible future scenarios, modelling incoming requests with a known probability distribution function. To optimise each deterministic scenario, the authors propose using a greedy heuristic to generate initial solutions, which are further improved using a local search algorithm. The experimental evaluation was performed over the same dataset used by Santos and Xavier (2013, 2015) from North-eastern Illinois, USA. A set of 12 different problem instances were generated, ranging from 134 to 758 passenger requests and 10 to 50 vehicles. The experimental results show that the proposed method is able to increase the number of served requests between 61 to 122%, when compared to an algorithm that does not take into account stochastic information of future requests. The solutions are computed in execution times that vary from 3 to 8 minutes approximately.

Several works have studied the taxi-sharing problem, both from the point of view of taxi companies that look forward to reduce their operational costs and from the point of view of passengers that want to save money by sharing their trips.

Balancing the interest of taxi owners and users, Ma, Zheng and Wolfson (2013) studied the dynamic taxi-sharing problem, where users' requests are scheduled in real time. Given a fix set of taxis and a flow of trip requests, the problem consist of defining which taxi should fulfill each new request in order to minimise the additional travelled distance. The proposed solution is a greedy strategy since it makes local decisions based on each new requests that arrives. To avoid computing the distance between each taxi and a new request's origin, the map is divided in a grid and the distances between each cell are pre-computed. This offers better performance at the expense of precision in the computed solutions. The proposed algorithm works as follows. First, a set of potential taxis to serve the request is generated based on the distance to the origin and the passenger's time windows. Then, the algorithm iterates through the set of taxis and inserts the trip in the schedule of the taxi that minimises the additional distance travelled. The experimental evaluation was performed using a data set of real taxi trajectories from Beijing, China. This data set with GPS information of more than 33,000 taxis collected for more than 87 days is also used in the experimental analysis of the algorithms proposed in this article. The results in a scenario with a proportion of 6 requests per taxi show that the greedy algorithm is able to fulfill 25% additional requests due to ride-sharing while minimising 13% the distance travelled by the set of taxis.

Closer to the problem being tackled which focusses on customers' interests, Tao and Chen (2007) presented two greedy heuristics to solve the taxi-sharing problem. It studies

both the one-origin-to-many-destinations problem and the inverse many-to-one version. The algorithm groups passengers according to their time windows. Additionally, each passenger can state a maximum number of passengers to share the ride with, and the gender preference for their potential travel partners. Each vehicle is considered to have a fixed capacity of four passengers. Optimisation of routes in order to minimise the travelled distance is performed once the passengers are already assigned to a taxi, by simply changing the relative order of the passengers in one taxi. This approach seems more naive than those proposed in other articles in the literature, since the amount of possible routes to explore with this approach is very limited. The experimental analysis was performed in a real-world taxi-sharing service comprised of 10 taxis and 798 passengers. Results showed an improvement in the travelled distance due to ride-sharing. However, the improvements are given in absolute terms not in percentage, so it is difficult to state the numeric efficacy of the proposed algorithm.

The analysis of related work indicates that the taxi-sharing problem is currently interesting in the scientific community, having a significant impact on both environment protection and economy. However, there are few user-oriented solutions in the literature, so there is still room to contribute in this line of research, by proposing efficient optimisation methods for planning and reducing vehicular traffic. Additionally, multiobjective approaches to solve the taxi-sharing problem are scarce. Multiobjective optimisation seems the right path to address the taxi-sharing problem, since the interest of a group of passengers willing to share rides may be very diverse and difficult to reflect in a single objective.

Our first work on the subject (Fagúndez, Massobrio and Nesmachnow, 2014) presented a single-objective variant of the taxi-sharing problem and proposed a simple EA that showed a good applicability for realistic medium-sized problem instances, but requiring large execution times. After that, a parallel EA was implemented to notably improve the quality of solutions found and the computational efficiency of the search (Massobrio, Fagúndez and Nesmachnow, 2014). Our first multiobjective approach to the problem applied a parallel micro EA using a linear aggregation approach to combine both objectives (Massobrio, Fagúndez and Nesmachnow, 2015a). Following this line of work, the MOEA presented at the 11th Metaheuristics International Conference (Massobrio, Fagúndez, and Nesmachnow, 2015b) was our first proposal for an explicit multiobjective algorithm to solve the problem. This article continues the work done on the multiobjective version of the taxi-sharing problem, by extending the experimental evaluation of the previously proposed algorithms using a new set of realistic problem instances.

4 Multiobjective evolutionary algorithms for taxi sharing

This section presents the details of the proposed MOEAs for the taxi-sharing problem.

4.1 Evolutionary algorithms and multiobjective optimisation

Evolutionary algorithms are non-deterministic metaheuristic methods that emulate the evolution of species in nature to solve optimisation, search, and learning problems (Bäck, Fogel and Michalewicz, 1997). In the past 30 years, evolutionary algorithms have been applied to solve many highly complex optimisation problems.

MOEAs have obtained accurate results when used to solve difficult real-life optimisation problems in many research areas (Deb, 2001; Coello, Van Veldhuizen and Lamont, 2002). Unlike many traditional methods for multiobjective optimisation, MOEAs

find a set with several solutions in a single execution, since they work with a population of tentative solutions in each generation.

MOEAs are designed taking into account two goals at the same time:

i approximating the Pareto front

ii maintaining diversity instead of converging to a reduced section of the Pareto front.

A Pareto-based evolutionary search leads to the first goal, while the second is accomplished using specific techniques that are also used in multimodal function optimisation (e.g., sharing, crowding).

The specific MOEAs used to solve the taxi-sharing problem are described below.

pμ-MOEA/D. *pμMOEA/D* combines two design features in order to solve a multiobjective optimisation problem:

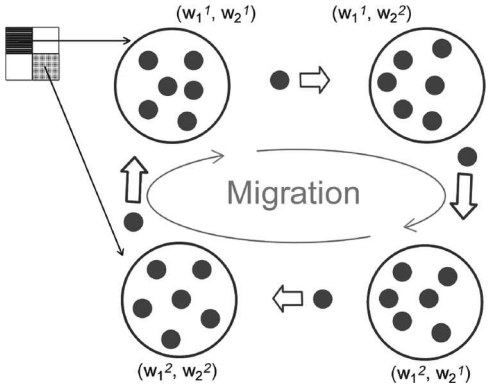
i A parallel approach following the distributed subpopulation (islands) model, using micropopulations and a migration operator to provide diversity.

ii A domain decomposition approach similar to the one applied in *MOEA/D* (Zhang and Li, 2007).

Each subpopulation in *MOEA/D* focusses in solving one specific optimisation problem, applying a linear aggregation of the objective functions using different weights for each objective, as described in the next paragraph.

In a problem with two objective functions f_1 and f_2, the fitness function is defined by the weighted sum of the objective functions of the problem, using a two-dimensional grid for the weights: $F_{i,j} = w_1^i \times f_1 + w_2^j \times f_2$, where $i \in \{1 \dots M\}, j \in \{1 \dots N\}$. If the grid is square and the weights are uniformly distributed among the islands, then the algorithm uses $M = N$ islands. Figure 1 shows the operation of *pμMOEA/D* for a two objective optimisation problem, where a square grid is used to define the weights.

Figure 1 pμ-MOEA/D schema



The approach using linear aggregation and domain decomposition in *pμMOEA/D* allows sampling the Pareto front of the optimisation problem even though the algorithm does not

use an explicit fitness assignment schema based on Pareto dominance. This approach has been successfully applied to the CHC algorithm before (Nesmachnow and Iturriaga, 2013).

NSGA-II. In this work, we apply the non-dominated sorting genetic algorithm (NSGA), version II (Deb, 2001), a state-of-the-art MOEA that has been successfully applied in many areas. NSGA-II has an improved evolutionary search function compared with its predecessor, NSGA, based on three features:

i a non-dominated, elitist ordering that diminishes the complexity of the dominance check

ii a crowding technique for diversity preservation

iii a fitness assignment method that considers crowding distance values.

NSGA-II classifies the population individuals into different *ranges*, formed by non-dominated solutions, with 0 being the best range (defined by the set of non-dominated solutions of the population). Within each range, solutions are ordered according to their *crowding distance* values, giving higher selection priority to those solutions with less density of neighbouring solutions.

4.2 Implementation details: encoding and objective functions

Solutions are represented as arrays of integers between 1 and N, representing the passengers, and $N-1$ zeros to separate passengers assigned to different taxis. The order for visiting the destinations is the one specified in the sequence. Figure 2 shows an example of the solution encoding for an instance with $N = 5$.

Figure 2 Example of solution representation for the taxi-sharing problem (see online version for colours)

The solution encoding has some restrictions/features:

i the number of consecutive (non-zero) integers is limited by the available taxi capacities as described in Section 2.2

ii each integer must appear only once in the encoding

iii consecutive zeros mean the same than a single one.

The objective functions to optimise according to a fully multiobjective approach are the total cost of the trips and the perceived delay as described in Eqs. (1) and (2).

4.3 Evolutionary operators

The proposed encoding has specific features and restrictions. Thus, we propose designing and applying ad hoc search operators conceived for the problem.

Population initialisation: the initial population is formed by the combination of two kind of solutions:

i Two individuals representing the greedy solutions for cost and delay (see a description of the greedy algorithms in Section 5.4.2).

ii The rest of the population is generated by applying a random number of perturbations (*i.e.*, swapping two elements) over copies of these two individuals.

The motivation for applying this seeding method for creating the initial population are related to the experimental results obtained for the single-objective version of the problem in Massobrio, Fagúndez and Nesmachnow (2014): in that work, the results indicated that using randomised greedy procedures in the initialisation allows significantly improving the quality of the evolutionary search in contrast to a full random initialisation.

Feasibility check and correction process: the initialisation method might violate some of the constraints defined for the solution encoding. Thus, a corrective function is applied to guarantee solution feasibility. The method searches for sequences of non-zero digits larger than the maximum vehicle capacity at the moment. Then, the correction algorithm locates the first consecutive couple of zeroes and moves the first zero to a random place at the non-zero streak, in order to break the invalid sequence. The search for invalid sequences continues until the end of the solution; at that point, the individual fulfills all problem constraints.

Selection: a *tournament selection*, is applied to provide an appropriate selection pressure. Two individuals are drawn from the population to compete with each other and the fittest one survives. Initial experiments confirmed that the standard proportional selection technique did not provide enough diversity, leading to premature convergence.

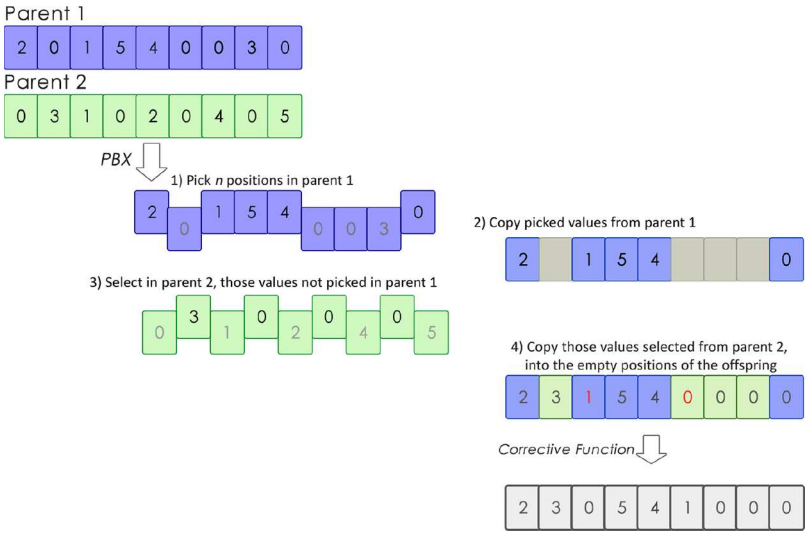
Recombination: we apply an ad hoc variant of the *position-based crossover* (PBX) operator, explained in Algorithm 1. In order to avoid violating the constraints imposed by the solution encoding, the same corrective function used after the initialisation is applied to the resulting offspring. Figure 3 shows an example of the PBX crossover for an instance with five passengers and with taxis with a maximum capacity of four available.

Algorithm 1 Ad hoc PBX for the taxi-sharing problem

1: Randomly select several positions in parent 1.
2: Partially generate the offspring, copying the selected values from parent 1.
3: Mark in parent 2 the positions already selected in parent 1.
4: Select the next non-marked value in parent 2, sequentially from the beginning, and copy it in the first free position in offspring.

Mutation: we apply the *Exchange Mutation* operator, which randomly selects and exchanges two positions in the solution encoding. Afterwards, the same corrective function mentioned above is applied to fix potential invalid solutions.

Figure 3 PBX applied to the taxi-sharing problem (see online version for colours)



Migration (pμ-MOEA/D): the migration operator is applied asynchronously, considering the islands connected in an unidirectional ring fashion. Emigrants are selected using a tournament selection (two individuals compete and the fittest one survives) and replace the worst individuals in the destination island. Migration acts as a diversity operator, avoiding premature convergence of the algorithm.

4.4 Implementing the final solution

When the final solution is computed, the set of passengers is split into groups that will travel in the same taxi. Since this assignment is guaranteed to be feasible (due to the correction process), there will be enough taxis available with the adequate capacities to fulfill the requests of the group of passengers. The assignments to the actual physical vehicles should be made by the users, accommodating the larger groups first to the vehicles that best fit their needs.

5 Experimental analysis

This section reports the experimental analysis of the proposed multiobjective EA to solve a set of realistic instances of the taxi-sharing problem.

5.1 Problem instances

A specific methodological approach for the generation of realistic problem instances was used, taking into account the problem restrictions and using services available to gather information about taxi demands, maps, and fares, including:

- *Taxi query generator (TQG)*, a tool that uses information from a database of taxi trajectories obtained from GPS devices installed for 87 days in 33,000 taxis in the city of Beijing, to generate realistic taxi demands. The data are a subset of those used

by Ma, Zheng and Wolfson (2013). TQG produces a list of origin/destination coordinates for individual trips. In order to design useful instances for the problem addressed in this article, we developed a script that groups trips that originate in nearby locations and generates one-origin-to-many-destinations problem instances.

- The *TaxiFareFinder* interface (TaxiFareFinder API, 2015), which was used to obtain the cost and trip-duration matrices for each generated instance of the problem, along with the prices corresponding to the minimum fare. Each pair of coordinates is sent to the interface TaxiFareFinder to get the cost and duration of the trip.

In response to reviewers' comments on our first article on the subject (Fagúndez, Massobrio and Nesmachnow, 2014), we also generated instances for the city of Montevideo, Uruguay. However, since taxi data are not available for the area, these instances were generated manually, by picking different points of interest in the city and choosing random destinations.

Following the proposed approach, we created a benchmark set of **88** realistic instances of the taxi-sharing problem, with different dimensions: *small* (10–15 passengers), *medium* (15–25 passengers), *large* (25–45 passengers), and a set of instances for the city of Montevideo, Uruguay (8–17 passengers). We defined different passengers' tolerances and vehicles' capacities for each instance.

5.2 Multiobjective optimisation metrics

A large number of metrics have been proposed in the literature to evaluate MOEAs (Coello, Van Veldhuizen and Lamont, 2002; Deb, 2001). In this work, we apply several of them to evaluate the results obtained from the proposed MOEAs in terms of convergence and correct sampling of the set of non-dominated solutions of the problem:

- The number of (different) *non-dominated solutions* (ND) computed by the algorithm.
- *Generational distance* (GD): the (normalised) sum of the distances between the non-dominated solutions in the Pareto front computed by the algorithm (solutions v in the approximated Pareto front P^*) and a set of uniformly distributed points in the true Pareto front [Eq. (3)]. Smaller values of GD mean a better approximation to the Pareto front.
- *Spacing*: evaluates the dispersion of non-dominated solutions in the calculated Pareto front [defined in Eq. (4)], where d_i is the distance between solution i in the computed Pareto front and its nearest neighbour, and $\bar{d}$ is the average of all d_i.
- Spread (s): evaluates the dispersion of non-dominated solutions in the computed Pareto front, including the distance from the extreme points of the true Pareto front [defined in Eq. (5)], where d_i is the distance between solution i in the computed Pareto front and its nearest neighbour, $\bar{d}$ is the average of all d_i, and d_h^e is the distance between the extreme of the h-th objective function in the true Pareto front and the closest point in the computed Pareto front (Deb, 2001). Smaller values of spread mean a better distribution of non-dominated solutions in the calculated Pareto front.
- *Hypervolume*: the volume (in the objective functions space) covered by the computed Pareto front.

- *Relative hypervolume* (RHV): the ratio between the volumes (in the objective functions space) covered by the computed Pareto front and the one covered by the true Pareto front. The ideal RHV value is 1.

$$GD = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|} \tag{3}$$

$$\sqrt{\frac{\sum_{i=1}^{ND} \left(\bar{d} - d_i\right)^2}{ND - 1}} \tag{4}$$

$$\frac{\sum_{h=1}^{k} d_h^e + \sum_{i=1}^{ND} \left|\bar{d} - d_i\right|}{\sum_{h=1}^{k} d_h^e + ND \times \bar{d}} \tag{5}$$

In Eqs. (4) and (5), d_i is the distance between the i-th solution in the computed Pareto front and its nearest neighbour, $\bar{d}$ is the average of all d_i, and d_h^e is the distance between the extreme of the h-th objective function in the true Pareto front and the closest point in the computed Pareto front.

The true Pareto front - which is unknown for the problem instances studied - is approximated by the set of non-dominated solutions found for each instance in each independent execution.

5.3 Parameter tuning

Given the stochastic nature of EAs, a parameter setting is mandatory prior to experimental analysis. For this purpose, a set of four medium-sized problem instances (different from those used in the experimental analysis to avoid bias) was generated following the method detailed in Section 5.1. The specific parameter tuning for each MOEA is specified below:

pμMOEA/D. After an initial evaluation, the size of the micropopulations was set to 15 individuals and the migration operator's frequency was set to 1,000 generations. The parameter tuning focussed on the crossover probability (p_C, candidate values 0.6, 0.75, and 0.95) and the mutation probability (p_M, candidate values 0.001, 0.01, and 0.1). Every combination of those values was tested, running 30 independent executions of 20,000 generations on each instance. The best results for the studied instances were obtained using $p_C = 0.75$ and $p_M = 0.1$.

NSGA-II. After initial experiments, the population size was set to 80 individuals. The parameter tuning focussed on the crossover probability (p_c, candidate values 0.6, 0.75, and 0.95) and the mutation probability (p_m, candidate values 0.001, 0.01, and 0.1). For each combination of candidate values, 30 independent executions were performed for each problem instance, with 5,000 generations on each run. Results suggest that using $p_C = 0.75$ and $p_M = 0.1$ allows finding the best solutions for the problem instances considered.

5.4 Experimental evaluation

The experimental analysis focussed on both the quality of the solutions and the performance of the proposed algorithms. For all 88 problem instances, 30 independent executions of 10,000 generations each were done.

5.4.1 Execution environment

The algorithms were executed at Cluster FING, which is the high-performance computing facility at Universidad de la República, Uruguay (Nesmachnow, 2010). The experimental analysis of the *pμMOEA/D* algorithm was done using 24 cores of a HP Proliant DL585 server comprised of AMD Opteron 6272 2.09 GHz processors with 48 GB of RAM available. The evaluation of the *NSGA-II* algorithm was performed on a HP Proliant DL385 G7 server, using a single core of an AMD Opteron 6172 2.10 GHz with 72 GB of RAM available.

5.4.2 Comparison against two greedy algorithms

A greedy algorithm is a method that constructs solutions by making locally optimal decisions in each step. In order to compare the results achieved by the proposed MOEAs, two greedy algorithms were developed for each of the objective functions: cost and delay. These greedy algorithms use ideas presented in the related works of Ma, Zheng and Wolfson (2013); Tao and Chen (2007) in order to group passengers according to their destinations and to assign them to the same taxis. Similar strategies can be expected from a group of human users trying to solve the problem manually.

The greedy method that minimises cost starts by adding the passenger closer to the origin in the first position of a newly formed taxi. Then, it continues adding passengers to that taxi sequentially, by picking the one with the destination closer to the one added in the previous step. The process continues until that taxi is full (in which case a new taxi is formed) or when all passengers are assigned to a taxi. An exception occurs when the cost of adding one passenger to the current taxi is greater than the one obtained by assigning a new taxi to serve that passenger's request. In that case, the current taxi is 'closed' and a new one is formed.

The greedy strategy for optimising delay takes the most hurried passengers (those with smaller tolerance for delay) and assigns each of them to a new taxi. It then takes the rest of the passengers in order of hurriedness and assigns them to the last location of the taxi that minimises their delay. When a taxi has as many passengers as the maximum capacity at the moment, it is consider as 'closed'. If one taxi has more passengers assigned than the maximum capacity available, it is deleted; and the passengers of that taxi are assigned to two new taxis.

5.4.3 Results and discussion

pμMOEA/D. Table 1 shows the results (grouped by family of instances) obtained by *pμMOEA/D* in the multiobjective metrics defined in Section 5.2. For each metric, the mean and standard deviation are reported and the best value is shown between brackets.

Table 1 Multiobjective metrics for *pμMOEA/D*

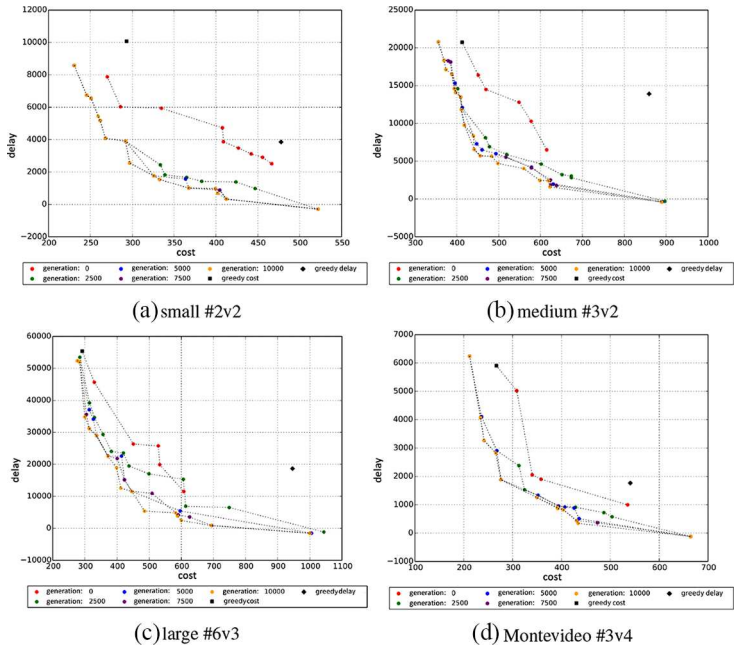
	#ND	*GD*	*Spacing*	*Spread*	*RHV*
Small	8.5 ± 2.1 (16.0)	3.1 ± 2.5 (0.0)	740.2 ± 746.3 (58.1)	0.6 ± 0.2 (0.1)	0.9 ± 0.1 (1.0)
Medium	9.1 ± 2.2 (19.0)	5.7 ± 2.5 (0.0)	1448.5 ± 1064.1 (141.6)	0.6 ± 0.1 (0.1)	0.9 ± 0.1 (1.0)
Large	8.5 ± 2.2 (17.0)	7.9 ± 3.4 (2.0)	2917.2 ± 2041.5 (175.3)	0.6 ± 0.1 (0.0)	0.8 ± 0.1 (1.0)
Montevideo	8.0 ± 2.1 (14.0)	3.0 ± 2.0 (0.0)	663.5 ± 542.4 (61.5)	0.6 ± 0.2 (0.0)	0.9 ± 0.0 (1.0)

Notes: GD, generational distance; RHV, relative hypervolume

The small GD values indicate good convergence towards the ideal Pareto front. Simultaneously, the low spread values indicate that *pμMOEA/D* maintains good diversity among the computed front of solutions. RHV values close to 1 reinforce these statements about convergence and diversity. However, the amount of non-dominated solutions seems low for the population size, suggesting that there is still room for improvement.

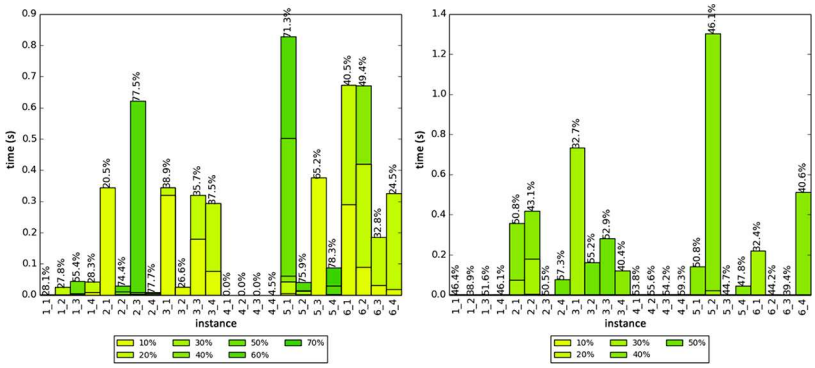
Figure 4 shows the solution front computed by *pμMOEA/D* every 2,500 generations and the solutions found by the greedy heuristics, on four representative problem instances. The fronts correspond to those executions that achieved the highest number of non-dominated solutions at the end of the run. It is worth noting the advance of the solution front towards a hypothetical Pareto front of the problem with the passing of generations. Moreover, a considerable improvement over the greedy algorithms can be observed.

Figure 4 *pμMOEA/D* - Pareto fronts with passing generations (see online version for colours)



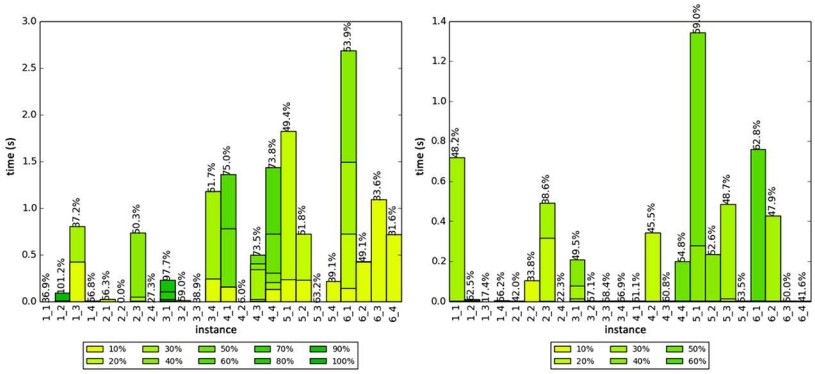
Figures 5 to 8 show the time needed by *pμMOEA/D* to outperform the solutions found by the greedy algorithms in each problem instance. The improvement over the greedy algorithm for cost represent the improvement in delay obtained by *pμMOEA/D*, in a solution with the same cost than the one found by that greedy algorithm. Analogously, the improvement over the greedy algorithm for delay express the improvement in cost obtained by *pμMOEA/D*, for a solution with the same delay that the one found by the greedy algorithm for delay. The time needed to outperform the greedy algorithms is shown with a step of 10% of improvement and, over each bar, the final improvement after 10,000 generations is indicated. The values shown correspond to those executions that achieved the highest improvement at the end of the execution.

Figure 5 *pμMOEA/D* - improvements over greedy heuristics on small instances (see online version for colours)



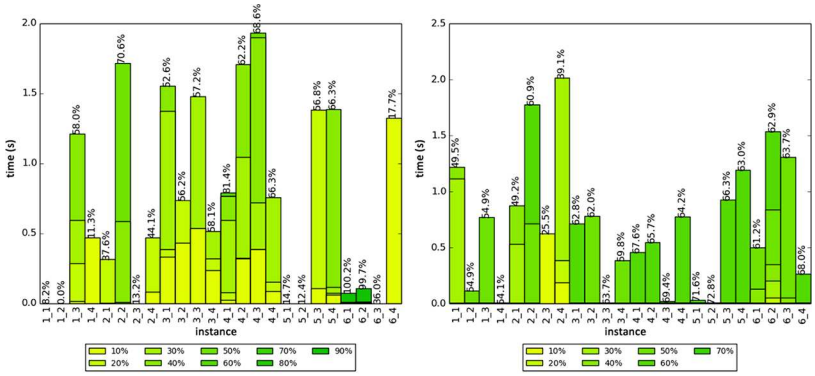
(a) cost (b) delay

Figure 6 *pμMOEA/D* - improvements over greedy heuristics on medium instances (see online version for colours)



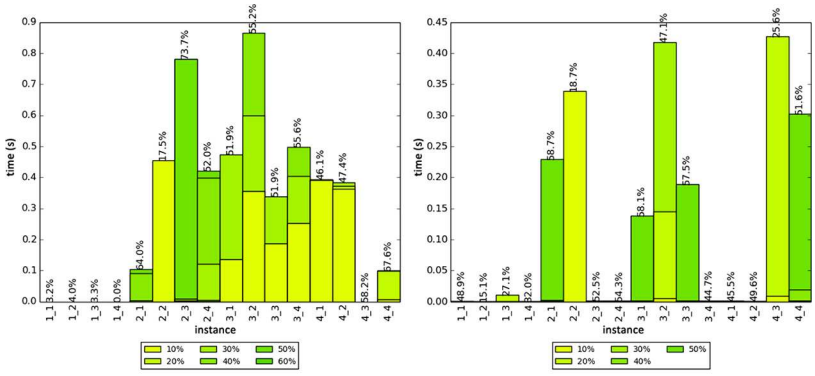
(a) cost (b) delay

Figure 7 *pμMOEA/D* - improvements over greedy heuristics on large instances (see online version for colours)



(a) cost (b) delay

Figure 8 *pμMOEA/D* - improvements over greedy heuristics on Montevideo instances (see online version for colours)



(a) cost (b) delay

pμMOEA/D is able to reach improvements over the solutions computed by the greedy algorithms in all problem instances. In the best case, it is able to outperform in up to **101.2%** the delay (101.2% on average), while keeping the same cost than the greedy algorithm for cost (medium instance #1v2); and improve up to **72.8%** the cost of the solution found by the greedy algorithm for delay (70.2% on average), while keeping the same delay (large instance #5v2). Moreover, the improvements are found, in most cases, in a few seconds after the start of the execution. This suggests the possibility of integrating the algorithm into a real-time user application.

NSGA-II. Table 2 shows the results achieved by *NSGA-II* in the multiobjective optimisation metrics described in Section 5.2. Results are grouped by family of instances. The mean and standard deviation are reported and the best value reached is shown between brackets.

Table 2 Multiobjective metrics for *NSGA-II*

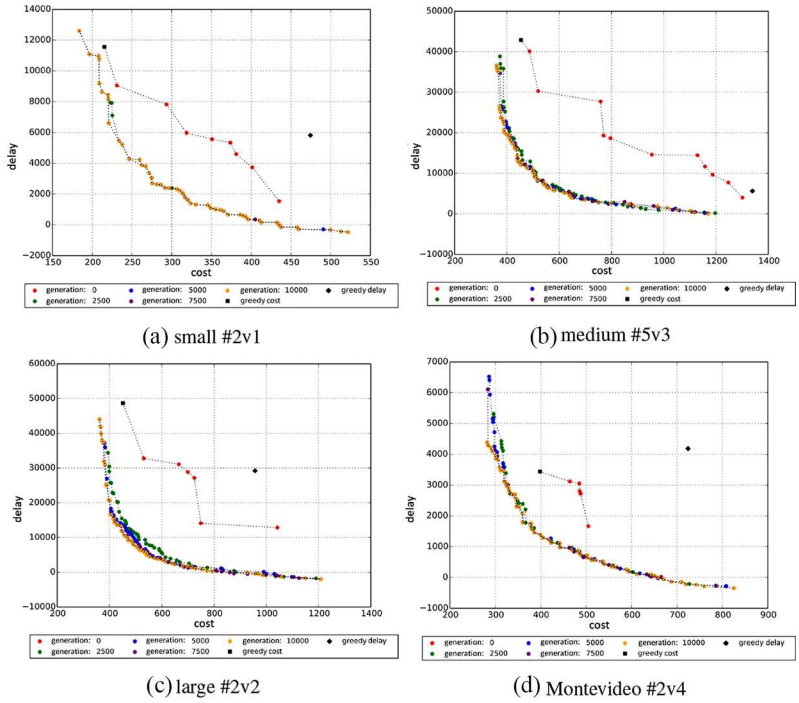
	#ND	*GD*	*Spacing*	*Spread*	*RHV*
Small	32.6 ± 9.5 (55.0)	0.3 ± 0.6 (0.0)	236.2 ± 222.7 (43.2)	0.9 ± 0.1 (0.7)	1.0 ± 0.0 (1.0)
Medium	54.5 ± 4.2 (67.0)	1.0 ± 0.7 (0.0)	193.6 ± 202.4 (26.2)	0.7 ± 0.2 (0.4)	1.0 ± 0.0 (1.0)
Large	55.2 ± 3.5 (67.0)	1.8 ± 1.1 (0.4)	243.6 ± 229.8 (26.4)	0.7 ± 0.2 (0.4)	1.0 ± 0.0 (1.0)
Montevideo	43.9 ± 16.4 (61.0)	0.4 ± 0.5 (0.0)	142.3 ± 143.2 (20.8)	0.8 ± 0.1 (0.5)	1.0 ± 0.0 (1.0)

Notes: GD, generational distance; RHV, relative hypervolume

It can be observed that the amount of non-dominated solutions found by *NSGA-II* is significantly larger than the one achieved by *pμMOEA/D*, reaching a peak of 67 non-dominated solutions in a population of size 80. The generational distance values are remarkably low, evidencing good convergence towards an ideal Pareto front. Spacing values reached by *NSGA-II* are significantly lower than those achieved by *pμMOEA/D*, suggesting a better diversity of solutions along the computed front. RHV values outperform those reached by *pμMOEA/D*, reinforcing the previous claims about convergence and diversity.

Figure 9 shows the Pareto fronts achieved by *NSGA-II* in four representative problem instances. The fronts shown correspond to those executions with the highest number of non-dominated solutions at the end of the run. The fronts are plotted every 2,500 generations along with the solutions found by the greedy heuristics.

Figure 9 *NSGA-II* - Pareto fronts with passing generations (see online version for colours)



(a) small #2v1 (b) medium #5v3

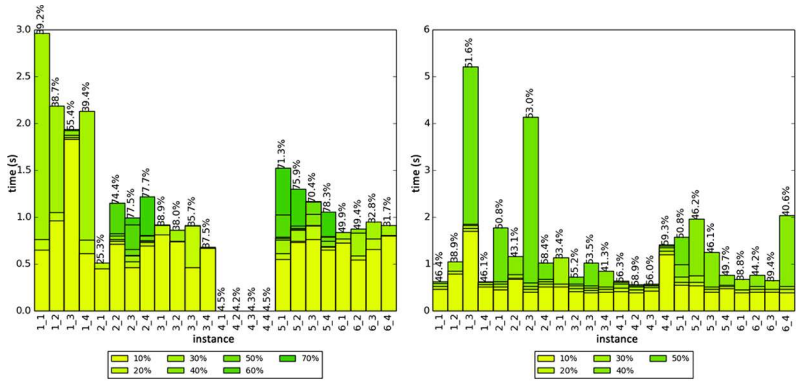
(c) large #2v2 (d) Montevideo #2v4

It can be seen how the computed fronts advance towards an ideal Pareto front with the passing of generations. It is also worth noting the significant improvements achieved by *NSGA-II* over both greedy algorithms. For small instance #2v1, *NSGA-II* does not show great differences with the passing of generations, suggesting that the algorithm is able to compute a good approximation to the ideal Pareto front early in the execution. However, on bigger instances, the solution fronts improve - even in small amounts - until the end of the execution. This raises the dilemma about the possibility of executing the algorithm for a longer number of generations, with the goal of obtaining better solutions. However, the algorithm is planned to be integrated in a web or mobile app for real users, requiring a reasonable response time that does not impact negatively in user's experience. For this reason, the strict limit of 10,000 generations as a stopping criterion is imposed, even at the expense of possible improvements in the results obtained.

Finally, Figures 10 to 13 show the necessary time to outperform the solutions found by the greedy algorithms in each problem instance. The improvement over the greedy algorithm for cost represents the improvement in delay obtained by *NSGA-II*, while keeping the same cost than the greedy algorithm for cost. The improvement over the greedy algorithm for delay represents the improvement in cost obtained by *NSGA-II* in a solution with the same delay than the the one achieved by the greedy algorithm for delay. Times are reported every 10% of improvement and the final improvement achieved after 10,000

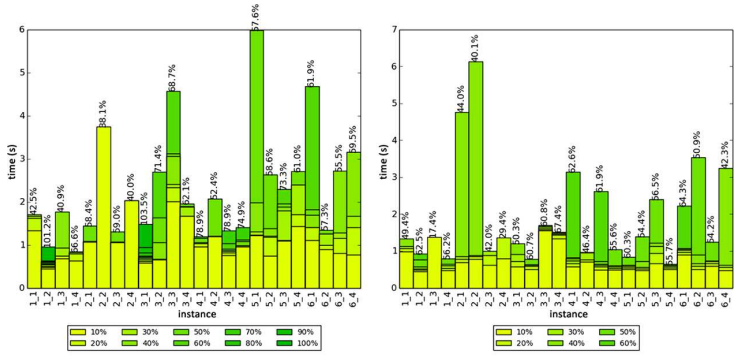
generations is indicated on the top of each bar. The reported values correspond to those executions that achieved the highest final improvement at the end of the execution.

Figure 10 *NSGA-II* - improvements over greedy heuristics on small instances (see online version for colours)



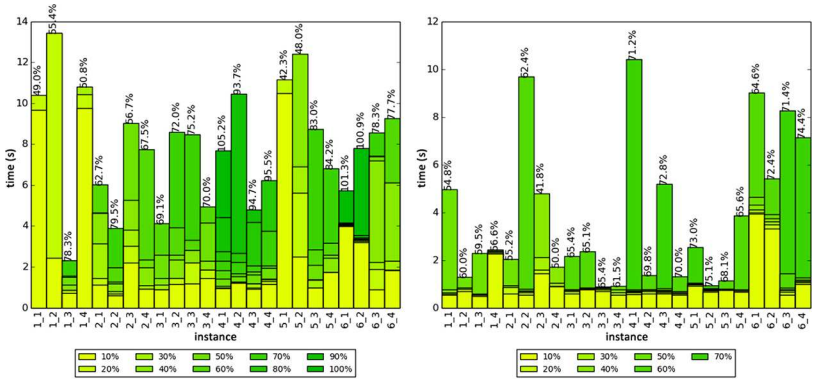
(a) cost (b) delay

Figure 11 *NSGA-II* - improvements over greedy heuristics on medium instances (see online version for colours)



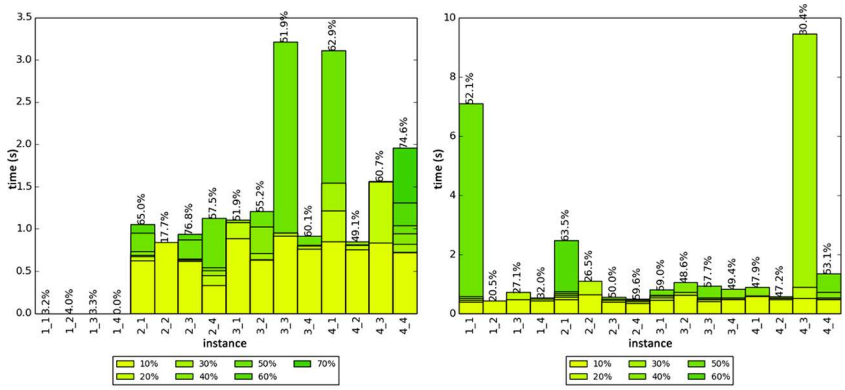
(a) cost (b) delay

Figure 12 *NSGA-II* - improvements over greedy heuristics on large instances (see online version for colours)



(a) cost (b) delay

Figure 13 *NSGA-II* - improvements over greedy heuristics on Montevideo instances (see online version for colours)



(a) cost (b) delay

NSGA-II is able to outperform both greedy algorithms in all problem instances studied. In the best case, it is able to outperform **105.2%** the delay (102.3% on average), while keeping the same cost than the greedy algorithm for cost (large instance #4v1), and improve up to **75.1%** the cost of the solution computed by the greedy algorithm for delay (73.0% on average), keeping the same delay (large instance #5v1). These improvements are achieved in a few seconds for small instances and in the order of 10 seconds for larger instances.

Comparative analysis. It is important to compare both algorithms in terms of quality of the solutions and computational efficiency.

Table 3 shows a comparative analysis of *pμMOEA/D* and *NSGA-II*. The improvements over each greedy algorithm are shown, along with the execution time in seconds and the hypervolume reached. The hypervolume is chosen to compare both algorithms as it takes into account both convergence and diversity. The mean, standard deviation, and the best values reported are computed considering the 30 independent executions of each problem instance.

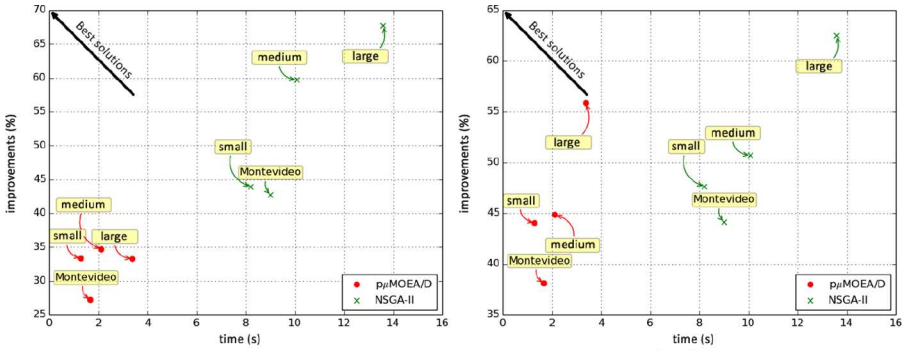
Table 3 Comparative table: *pμMOEA/D* vs. *NSGA-II*

		Improvement over greedy (%)			
		Cost	*Delay*	*Time (s)*	*HV* ($\times 10^6$)
Small	pμMOEA/D	33.3 ± 24.3 (78.3)	44.0 ± 7.8 (59.3)	1.3 ± 0.4 (0.7)	1.2 ± 1.3 (4.8)
	NSGA-II	43.9 ± 24.3 (78.3)	47.6 ± 7.3 (59.3)	8.2 ± 0.4 (7.3)	1.3 ± 1.4 (5.2)
Medium	pμMOEA/D	34.7 ± 27.3 (101.2)	44.8 ± 15.0 (66.9)	2.1 ± 0.7 (1.1)	5.8 ± 4.3 (19.3)
	NSGA-II	59.7 ± 19.2 (103.5)	50.7 ± 11.6 (67.4)	10.1 ± 0.9 (8.3)	7.2 ± 5.3 (23.2)
Large	pμMOEA/D	33.3 ± 27.1 (100.2)	55.8 ± 12.5 (72.8)	3.4 ± 1.2 (1.9)	10.0 ± 8.9 (39.2)
	NSGA-II	67.7 ± 23.9 (105.2)	62.5 ± 8.3 (75.1)	13.6 ± 1.8 (11.6)	13.2 ± 10.0 (43.1)
Montevideo	pμMOEA/D	27.2 ± 20.2 (73.7)	38.1 ± 16.6 (58.7)	1.7 ± 0.6 (0.8)	2.5 ± 1.8 (6.5)
	NSGA-II	42.7 ± 26.3 (76.8)	44.1 ± 13.9 (63.5)	9.0 ± 0.8 (7.3)	2.9 ± 2.1 (7.1)

The Shapiro-Wilk test (with a confidence level of 95%) performed over the values of hypervolume allows rejecting (in a considerable number of problem instances) the null hypothesis that the hypervolume values reached over the 30 independent executions follow a normal distribution. Therefore, the Kruskal-Wallis test was applied over the hypervolume values in order to compare the results obtained by *pμMOEA/D* against those computed by *NSGA-II*. The results of the Kruskal-Wallis test suggest that there is statistical difference between the hypervolumes achieved by both algorithms in all problem instances studied. Since the mean hypervolume values achieved by *NSGA-II* outperform those achieved by *pμMOEA/D* in all problem instances, it can be stated that the former algorithm outperforms the latter. In large instances, the improvement over the greedy algorithm for cost in the best case was of 105.2% for *NSGA-II* vs. 100.2% for *pμMOEA/D* (67.7 vs. 33.3% on average). The improvement over the greedy algorithm for delay was 75.1% for *NSGA-II* vs. 72.8% for *pμMOEA/D* in the best case (62.5 vs. 55.8% on average).

Figure 14 reports the trade-off between the improvements achieved over the greedy algorithms and the execution times of both MOEAs. Results are grouped by family of instances and correspond to the mean values achieved over the 30 independent executions.

Figure 14 *pμMOEA/D* and *NSGA-II* - improvement over greedys vs. execution time (see online version for colours)

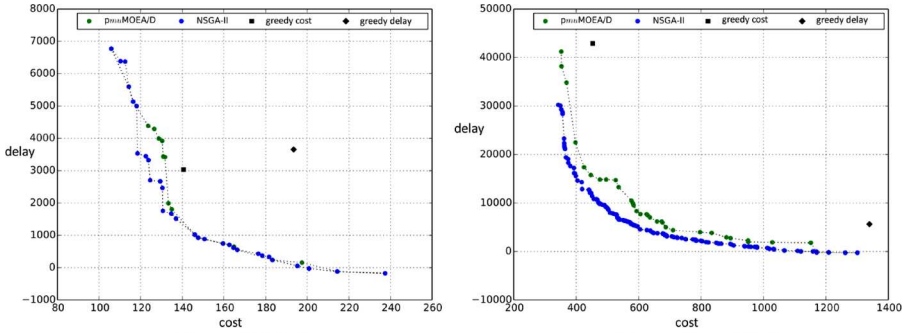


(a) cost (b) delay

The explicit multiobjective approach of *NSGA-II* allows reaching better results that the domain decomposition strategy of *pμMOEA/D*. However, execution times for *pμMOEA/D* are significantly lower than the ones needed by *NSGA-II*. It is precise to highlight that *pμMOEA/D* uses multiple cores, while *NSGA-II* was executed using a single core. For an online application, it is necessary to compute solutions in reduced execution times. Therefore, one of the main lines of future work is to develop a variant of the *NSGA-II* algorithm following the distributed subpopulation model. The design of a solution that combines both approaches needs to take into consideration that the computation of the crowding distance is done over the whole population. Therefore, it is necessary to either include a centralised mechanism to compute crowding distance or change the mechanism to preserve diversity.

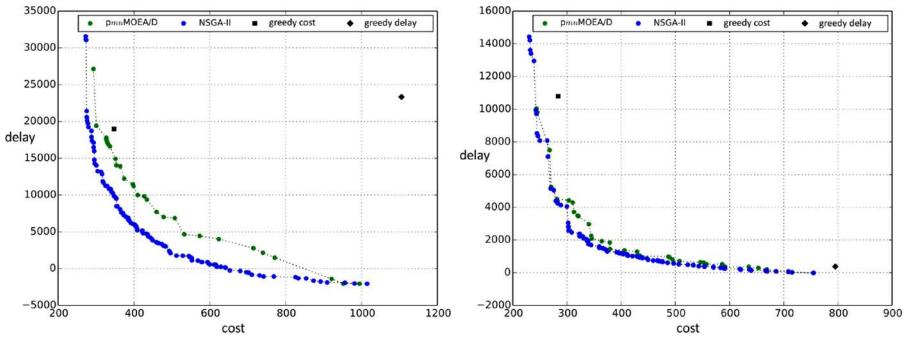
Figures 15 and 16 show the fronts computed by both MOEAs and the solutions found by the greedy heuristics for one representative instance of each family. The fronts shown are global, in the sense that they are formed combining all the non-dominated solutions found over the 30 independent executions of each MOEA.

Figure 15 *pμMOEA/D* vs. *NSGA-II* - Pareto fronts for small and medium instances (see online version for colours)



(a) small #6v1 (b) medium #5v3

Figure 16 *pμMOEA/D* vs. *NSGA-II* - Pareto fronts for large and Montevideo instances (see online version for colours)



(a) large #5v2 (b) Montevideo #4v3

Both MOEAs outperform the solutions found by the greedy algorithms. *NSGA-II* achieves better solutions, particularly in large problem instances. Moreover, the amount of non-dominated solutions found by *NSGA-II* is significantly higher than *pμMOEA/D* and those solutions are better spread along the front.

6 Conclusion and future work

This article studied the multiobjective taxi-sharing problem from the point of view of taxi users. A specific problem formulation is presented, considering the total cost and total delay objectives, and two MOEAs are applied to find accurate solutions for a set of realistic problem instances: one following a domain decomposition approach and one with an explicit multiobjective Pareto-based fitness assignment. The experimental analysis demonstrates the applicability of the studied MOEAs to find a wide set of trade-off solutions for the problem, considering different values for the problem objectives. Experimental results show that *NSGA-II* achieved better results than *pμMOEA/D* on all problem instances studied. *NSGA-II* was able to improve up to **105.2%** the delay achieved by the greedy algorithm for cost, while keeping the same cost; and improved up to **75.1%** the cost achieved by the greedy algorithm for delay, maintaining the same delay. However, execution times for *pμMOEA/D* are significantly lower than those measured for

NSGA-II. Therefore, it is necessary to set a compromise level between quality of solutions and response time.

The main lines of future work include adding real-time traffic information and taxi availability to the system, in order to consider delays and alternative routes. Additionally, the design of a variant of the *NSGA-II* algorithm that considers the subpopulation models is considered, in order to obtain both the good results associated with the explicit multiobjective approach and the reduced execution times thanks to the island model.

Problem instances and detailed results are available at www.fing.edu.uy/inco/grupos/cecal/hpc/AG-Taxi/. A prototype online application for end users (considering only the cost objective) is available at www.mepaseaste.uy.

References

Bäck, T., Fogel, D. and Michalewicz, Z. (Eds.) (1997) *Handbook of Evolutionary Computation*, Oxford University Press, Oxford.

Coello, C., Van Veldhuizen, D. and Lamont, G. (2002) *Evolutionary Algorithms for Solving Multi-objective Problems*, Kluwer, New York.

Cordeau, J-F. and Laporte, G. (2003) 'A tabu search heuristic for the static multi-vehicle dial-a-ride problem', *Transportation Research Part B: Methodological*, Vol. 37, No. 6, pp.579–594.

Deakin, M. and Al Waer, H. (2011) 'From intelligent to smart cities', *Intelligent Buildings International*, Vol. 3, No. 3, pp.133–139.

Deb, K. (2001) *Multi-objective Optimisation Using Evolutionary Algorithms*, John Wiley & Sons, Chichester.

Fagúndez, G., Massobrio, R. and Nesmachnow, S. (2014) 'Resolución en línea del problema de viajes compartidos en taxis usando algoritmos evolutivos', *Proceedings of the Latin American Computing Conference*, Montevideo, Uruguay.

Fellows, N. and Pitfield, D. (2000) 'An economic and operational evaluation of urban car-sharing', *Transportation Research Part D: Transport and Environment*, Vol. 5, No. 1, pp.1–10.

Ferrari, E., Manzini, R., Pareschi, A., Persona, A. and Regattieri, A. (2003) 'The car ing problem: heuristic algorithms based on savings functions', *Journal of Advanced Transportation*, Vol. 37, pp.243–272.

Hartman, I.B-A., Keren, D., Abu Dbai, A., Cohen, E., Knapen, L., Yasar, A-U-H. and Janssens, D. (2014) 'Theory and practice in large carpooling problems', *Procedia Computer Science*, Vol. 32, pp.339–347.

Hopcroft, J.E. and Karp, R.M. (1971) 'A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs', *12th Annual Symposium on Switching and Automata Theory*, East Lansing, USA, pp.122–125.

Katzev, R. (2003) 'Car sharing: a new approach to urban transportation problems', *Analyses of Social Issues and Public Policy*, Vol. 3, No. 1, pp.65–86.

Knapen, L., Hartman, I.B-A., Keren, D., Yasar, A-U-H., Cho, S., Bellemans, T., Janssens, D. and Wets, G. (2015) 'Scalability issues in optimal assignment for carpooling', *Journal of Computer and System Sciences*, Vol. 81, No. 3, pp.568–584.

Letchford, A., Eglese, R. and Lysgaard, J. (2002) 'Multistars, partial multistars and the capacitated vehicle routing problem', *Mathematical Programming*, Vol. 94, No. 1, pp.21–40.

Ma, S., Zheng, Y. and Wolfson, O. (2013). 'T-share: A large-scale dynamic taxi ridesharing service', *IEEE 29th International Conference on Data Engineering*, Brisbane, Australia, pp.410–421.

Manna, C. and Prestwich, S. (2014) 'Online stochastic planning for taxi and ridesharing', *IEEE 26th International Conference on Tools with Artificial Intelligence*, Limassol, Cyprus, pp.906–913.

Massobrio, R., Fagúndez, G. and Nesmachnow, S. (2014) 'A parallel micro evolutionary algorithm for taxi sharing optimization', *VII ALIO/EURO Workshop on Applied Combinatorial Optimization*.

Massobrio, R., Fagúndez, G. and Nesmachnow, S. (2015a) 'Planificación multiobjetivo de viajes compartidos en taxis utilizando un micro algoritmo evolutivo paralelo', *X Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, Mérida-Almendralejo, Spain.

Massobrio, R., Fagúndez, G. and Nesmachnow, S. (2015b) 'Multiobjective taxi sharing optimization using the NSGA-II evolutionary algorithm', *11th Metaheuristic International Conference*, Agadir, Morocco.

Nesmachnow, S. (2010) 'Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República', *Revista de la Asociación de Ingenieros del Uruguay*, Vol. 61, pp.12–15.

Nesmachnow, S. and Iturriaga, S. (2013) 'Multiobjective grid scheduling using a domain decomposition based parallel micro evolutionary algorithm', *International Journal of Grid and Utility Computing*, Vol. 4, pp.70–84.

Pfanner, E. (2012) 'Ride-sharing services grow popular in Europe', *The New York Times*, October 1, 2012.

Santos, D.O. and Xavier, E.C. (2013) 'Dynamic taxi and ridesharing: a framework and heuristics for the optimization problem', *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, Beijing, China, pp.2885–2891.

Santos, D.O. and Xavier, E.C. (2015) 'Taxi and ride sharing: a dynamic dial-a-ride problem with money as an incentive', *Expert Systems with Applications*, Vol. 42, No. 19, pp. 6728–6737.

Tao, C-C. and Chen, C-Y. (2007) 'Heuristic algorithms for the dynamic taxipooling problem based on intelligent transportation system technologies', *4th International Conference on Fuzzy Systems and Knowledge Discovery*, Haikou, China, pp.590–595.

TaxiFareFinder API (2015) TaxiFareFinder API, Website, http://www.taxifarefinder.com (last accessed April 26, 2016).

Zhang, Q. and Li, H. (2007) 'MOEA/D: a multiobjective evolutionary algorithm based on decomposition', *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 6, pp.712–731.