Multiobjective taxi sharing optimization using the NSGA-II evolutionary algorithm

Renzo Massobrio, Sergio Nesmachnow, Gabriel Fagúndez

Universidad de la República, Uruguay
{renzom,sergion,gabrielf}@fing.edu.uy

Abstract

This article presents the application of the NSGA-II multiobjective evolutionary algorithm to the problem of distributing passengers traveling 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 the users: minimize the total cost of the trips and the time of travel for each passenger from the origin to its destination. The proposed method follows a fully multiobjective approach, and it is designed to provide an accurate and efficient way to solve realistic instances of the problem with high practical applicability. The experimental analysis compares the solutions found using the proposed algorithm versus those computed using a previous parallel micro evolutionary algorithm (following a linear aggregation approach to combine the problem objectives) and two greedy heuristics. The results show that the multiobjective evolutionary algorithm is able to efficiently reach significant improvements in both problem objectives in short execution times.

1 Introduction

Car pooling is the concept of sharing vehicles among people with similar travel needs. The idea has gained massive public attention in recent years [6]. Using shared vehicles has both economical and environmental benefits, at individual and collective levels. Having less vehicles on the streets leads to fewer traffic jams, resulting in a more fluent, thus more efficient traffic. This contributes to minimizing the impact of transportation in the environment, which is a major concern nowadays, specially for big cities [7] [8]. From an economical perspective, sharing trips between multiple passengers significantly reduces transportation costs.

Taxis are a fast and reliable mean of transportation. However, they rarely run at full capacity, and they could therefore benefit from the car pooling idea. There are some web platforms that provide solutions for ride-sharing scheduling, and some research works on different variants of the *taxi pooling* problem. The taxi sharing problem is NP-hard [9]. Thus, heuristics and metaheuristics [16] are needed to find high-quality solutions in reasonable execution times for realistic problem instances.

This article presents a multiobjective evolutionary algorithm (MOEA) to solve the multi-objective one-origin-multiple-destinations variant of the taxi sharing problem. The experimental evaluation over real-world scenarios demonstrates that the proposed method is an accurate and efficient tool to solve the problem, which can be easily integrated in on-line (web, mobile) applications.

The article is organized as follows. Section 2 introduces the multiobjective version of the taxi sharing problem and reviews related work on the topic. Section 3 introduces evolutionary algorithms (EAs) and their multiobjective variants, and describes the proposed MOEA to solve the problem. Section 4 reports the experimental evaluation, including a comparison against a previous EA applying domain decomposition and linear aggregation of objectives from [14] and two greedy heuristics to solve the problem. Finally, Section 5 formulates the conclusions and the main lines for future work.

2 The multiobjective taxi sharing problem

This section introduces the taxi sharing problem and its mathematical formulation as a multiobjective optimization problem, and reviews related works on the topic.

2.1 Problem description and model

The *taxi pooling* problem models a reality where a number of people located in the same origin, decide to travel to different destinations using taxis. The optimization problem proposes finding an appropriate number of taxis and the distribution of passengers, with the goal of simultaneously minimizing two important objectives for users: the cost of the travel and the perceived delay to reach their destinations. This 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 problem, the following considerations should be taken into account:

- 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 to the algorithm.
- The maximum number of taxis that can be used in a scenario with N passengers is N, in the case where every passenger travels in a separate vehicle.
- The cost for each taxi includes the cost to hire the taxi (minimum fare), and the cost for traveling 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 journey from the origin to their destinations. The tolerance of each passenger is also given as an input to the algorithm.

2.2 Mathematical formulation

The mathematical formulation is presented below.

Given the following elements:

- A set of passengers $P = \{p_1, p_2, \ldots, p_N\}$ starting from a common geographic point O and having a set of potentially different destinations $D = \{d_1, d_2, \ldots, d_N\}$.
- 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 that determines how many passengers use a taxi on a given trip, where $C_i \leq K_i$.
- A matrix CM of dimension $(N+1) \times (N+1)$ specifying the cost between each of the geographic locations involved in the problem (an origin point and destination points N).
- A matrix TM of dimension $(N+1) \times (N+1)$ that specifies the traveling time between each of the geographic locations.
- A vector T of length N where T_i is the extra time that user i tolerates over the time it takes to go directly from the origin O to destination d_i.
- A *total cost function*, determined by the minimum fare (MF) and the distance cost from the origin to the final destination:

$$TC = \sum_{t_i} (MF + \sum_{j=1}^{C(t_i)} CM[d_{j-1}, d_j]) \tag{1}$$

- A *total delay function*, determined by the difference between the time tolerated and the actual time that each passenger takes to reach its destination:

$$TD = \sum_{t_i} \left(\sum_{j=1}^{C(t_i)} \left(T_i + TM[0, d_j] - \left(\sum_{h=1}^{j} (TM[h-1, h] \right) \right) \right) \tag{2}$$

The problem aims to find a planning function $f{:}P{\rightarrow}T$ to transport the N passengers in K taxis ($K{\leq}M$), determining both the assignment of passengers to taxis and the order to visit the destinations, simultaneously minimizing the total cost (Eq. 1) and the total delay (Eq. 2).

2.3 Related work

The *taxi pooling* problem is a variant of the *car pooling* problem, which has been studied from different perspectives in the related literature.

From the point of view of taxi companies, Xin *et al*. [18] optimized the cost of k taxis serving clients on-line, applying a heuristic that sends the two nearest taxis to attend a request, promoting competition and avoiding underutilization. The results showed that the problem holds a competitive ratio of k against an optimal algorithm that knows the entire sequence of requests beforehand.

Balancing the interest of taxi owners and users, Ma *et al*. [12] proposed a dynamic system for taxi sharing, by combining a search and a planning method to find taxis and assign passengers. A lazy strategy is applied to improve execution times, delaying the shortest path calculation as much as possible. Using a database of 33.000 GPS taxi trajectories from Beijing, the dynamic system reduced the travelled distances up to 13%, while serving 25% more requests in a simulated scenario with 6 requests per taxi.

Closer to the problem being tackled which focuses on customers' interests, Tao *et al*. [17] proposed two heuristic algorithms to optimize taxi ride-sharing costs, based on greedy strategies for the one-origin-to-many-destinations and for the many-to-one scenarios. The results of a field test of taxi-pooling at Taipei with 10 taxis and 798 passengers showed an average matching success rate of 60.3%. However, the fuel savings results are shown only in absolute terms, making it difficult to extract meaningful information to compare with other techniques. We address the one-to-many problem variant, so it is of particular interest to see the performance of the greedy method against our proposed MOEA.

From the point of view of multiobjective optimization, Lin et al. [10] proposed a Simulated Annealing approach to minimize the operational cost for a taxi fleet and maximize the user satisfaction. The problem formulation takes into account the distance traveled by each vehicle and the waiting time of clients. The proposed algorithm was able to reduce the distances in 19% and increasing by 66% the number of taxis available, when compared with a traditional scenario that does not apply taxi sharing.

The analysis of related works indicates that the taxi sharing problem is currently interesting for the scientific community, having a significant impact on both environment protection and economy. However, there are few user-oriented solutions in literature, so there is still room to contribute in this line of research, by proposing efficient optimization methods for planning and reducing vehicular traffic.

Our previous work [5] proposed a simple EA for taxi sharing optimization that showed a good applicability for realistic medium-size problem instances, but requiring large execution times. After that, a parallel EA was implemented to notably improve the results quality and the the computational efficiency of the search [13]. Our first multiobjective approach to the problem applied a parallel micro EA using a linear aggregation approach to combine both objectives [14]. Following this line of work, the MOEA we introduce in this article is the first proposal for an explicit multiobjective algorithm to solve the problem.

3 A multiobjective evolutionary algorithm for taxi sharing

This section presents details of the proposed multiobjective evolutionary algorithm for taxi sharing optimization and the greedy heuristics used as a baseline for the results comparison.

3.1 Evolutionary algorithms and NSGA-II

Evolutionary algorithms are non-deterministic metaheuristic methods that emulate the evolution of species in nature to solve optimization, search, and learning problems [2]. In the past 30 years, evolutionary algorithms have been applied to solve many highly complex optimization problems.

MOEAs [3, 4] have obtained accurate results when used to solve difficult real-life optimization problems in many research areas. Unlike many traditional methods for multiobjective optimization, MOEAs find a set with several solutions in a single execution, since they work with a population of solutions in each generation. MOEAs are designed taking into account two goals at the same time: i) approximating the Pareto front, and 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 multi-modal function optimization (e.g. sharing, crowding).

In this work, we apply the non-dominated sorting genetic algorithm (NSGA), version II [4], 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; and iii) a fitness assignment method that considers crowding distance values. The algorithm was developed using ECJ [11], a Java-based evolutionary computation research system.

3.2 Implementation details: encoding and objective functions

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

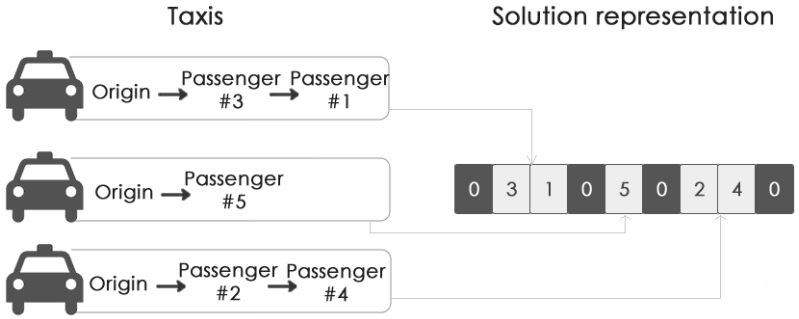


Figure 1: Example of solution representation for the taxi sharing problem.

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 optimize according to a fully multiobjective approach are the total cost of the trips and the perceived delay as described in Eq. (1) and Eq. (2).

3.3 Evolutionary Operators

The proposed encoding has specific features and restrictions, so we apply ad-hoc search operators.

Population initialization: the initial population consists of two individuals representing the Greedy solutions for cost and delay (see Section 4.4.1) and 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.

Feasibility check and correction process: the initialization 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 sequence, 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* (tournament size: 2 individuals) is applied to provide an appropriate selection pressure. 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 initialization is applied to the resulting offsprings. Figure 2 shows an example of the PBX crossover for an instance with 5 passengers.

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.

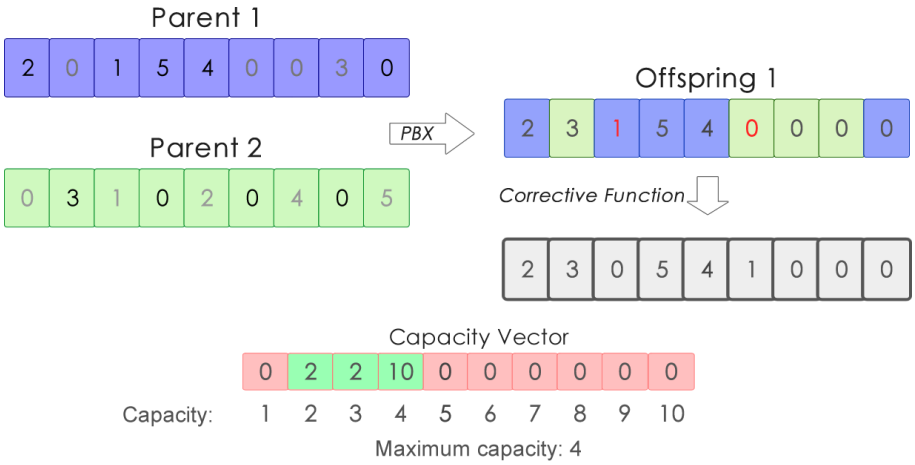


Figure 2: PBX applied to the taxi sharing problem.

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

4 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.

4.1 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 a week in 10,357 taxis in the city of Beijing, to generate realistic taxi demands. The data are a subset of those used by Ma et al. [12]. 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 [1], which was used to obtain the cost matrix for each generated instance of the problem, along with the prices corresponding to the minimum fare. Each pair of coordinates for a given instance of the problem is sent to the interface TaxiFareFinder to get the cost between each point.

Following the proposed approach, we created a benchmark set of **41** realistic instances of the taxi sharing problem, with different dimensions: *small* (9–24 passengers), *medium* (26–37 passengers), *large*

(42–53 passengers) and a set of instances for the city of Montevideo, Uruguay (9–44 passengers), considering the origins in different points of interest in the city and choosing random destinations. For each instance, we defined different passengers tolerances and vehicles capacities.

4.2 Multiobjective optimization metrics

A large number of metrics have been proposed in the literature to evaluate MOEAs [3, 4]. In this work, we apply several of them to evaluate the results obtained from the NSGA-II algorithm, convergence and correct sampling of the set of non-dominated solutions of the problem:

- The number of (different) non-dominated solutions (ND).
- Generational Distance (GD): the (normalized) 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 (Eq. 4).
- 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 (Eq. 5). Smaller values of spread mean a better distribution of non-dominated solutions.
- 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 true Pareto front. The ideal RHV value is 1.

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

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

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

In Eq. 4 and 5, d_i is the distance between the i-th solution in the computed Pareto front and its nearest neighbor, $\bar{d}$ 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 execution of the proposed MOEA.

4.3 Parameter tuning

Given the stochastic nature of EAs a parameter setting analysis in mandatory prior to experimental analysis. For this purpose, a set of 4 medium-sized problem instance (different from those used in the experimental analysis in order to avoid bias) was generated following the method detailed in Section 4.1.

After initial experiments the population size was set to 80 individuals. The parameter tuning focused 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 5000 generations on each run.

The parameter settings results suggest that using $p_c = 0.75$ and $p_m = 0.1$ allows computing the best results for the problem instances used.

4.4 Experimental evaluation

The experimental analysis focused on both the quality of the solutions and the performance of the proposed NSGA-II algorithm. For all 41 problem instances 30 independent executions of 5000 generations each were done. The experimental analysis was performed on an Intel Core i7-4500U CPU @ 1.80GHz, 8GB of RAM with 64 bit Ubuntu 14.10.

4.4.1 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 NSGA-II, two greedy algorithms were developed for each of the objective functions: cost and delay. Similar strategies can be expected from a group of human users trying to solve the problem.

The greedy method that minimizes cost sequentially adds the passenger whose destination is closer to the origin in a taxi, until that taxi is full. In the case where 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 request, the current taxi is "closed", and a new one is formed. The algorithm ends when every passenger is assigned to a taxi.

The greedy strategy for optimizing the 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 minimizes 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.

4.4.2 Results and discussion

The experimental results of the NSGA-II were compared to those achieved by the pμ-MOEA/D previously developed by our research group [14]. Table 1 shows the improvement against the greedy strategies, the execution time in seconds and the hypervolume for both algorithms.

Results show that the NSGA-II is able to improve the greedy algorithm for cost in up to 69.9% in the best case (21.2% on average) for large instances and the greedy algorithm for delay in up to 135.9% (120.2% on average) for medium instances.

Furthermore, the NSGA-II is able to get better improvements than the pμ-MOEA/D in shorter execution times. It is important to highlight that the pμ-MOEA/D was executed at the Cluster Fing high performance computing facilty [15] using 24 cores. Moreover, the NSGA-II achieved better hypervolumes values (both in best case and average) indicating a good approximation to the ideal pareto front while keeping good diversity.

Table 2 shows the multiobjective metrics values for the NSGA-II as described in Section 4.2. The number of non-dominated points on the final generation is equal to the population size for all instances. The values of generational distance are small, indicating good convergence to the pareto front. The small spread values obtained are evidence of good diversity in the solutions, and the relative hypervolumes values close to 1 indicate good coverage and convergence to the pareto front.

Figure 3 shows the results achieved by the NSGA-II, the pμ-MOEA/D and the greedy algorithms for one representative instance of each class. The NSGA-II outperforms the pμ-MOEA/D and greedy algorithms, and is able to reach a greater number of non-dominated solutions closer to the pareto front.

5 Conclusions 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 the NSGA-II multiobjective evolutionary algorithm is applied to find accurate solutions for a set of realistic problem instances.

Unlinke our previous work [14], the problem is solved following a explicit multiobjective approach, instead of using a linear aggregation function and weights for the problem objectives.

The experimental analysis demonstrates the applicability of the studied MOEA to find a wide set of trade-off solutions for the problem, considering different values for the problem objectives. The numerical results report that the NSGA-II is able to outperform a greedy heuristic for cost in up to 69.9% in

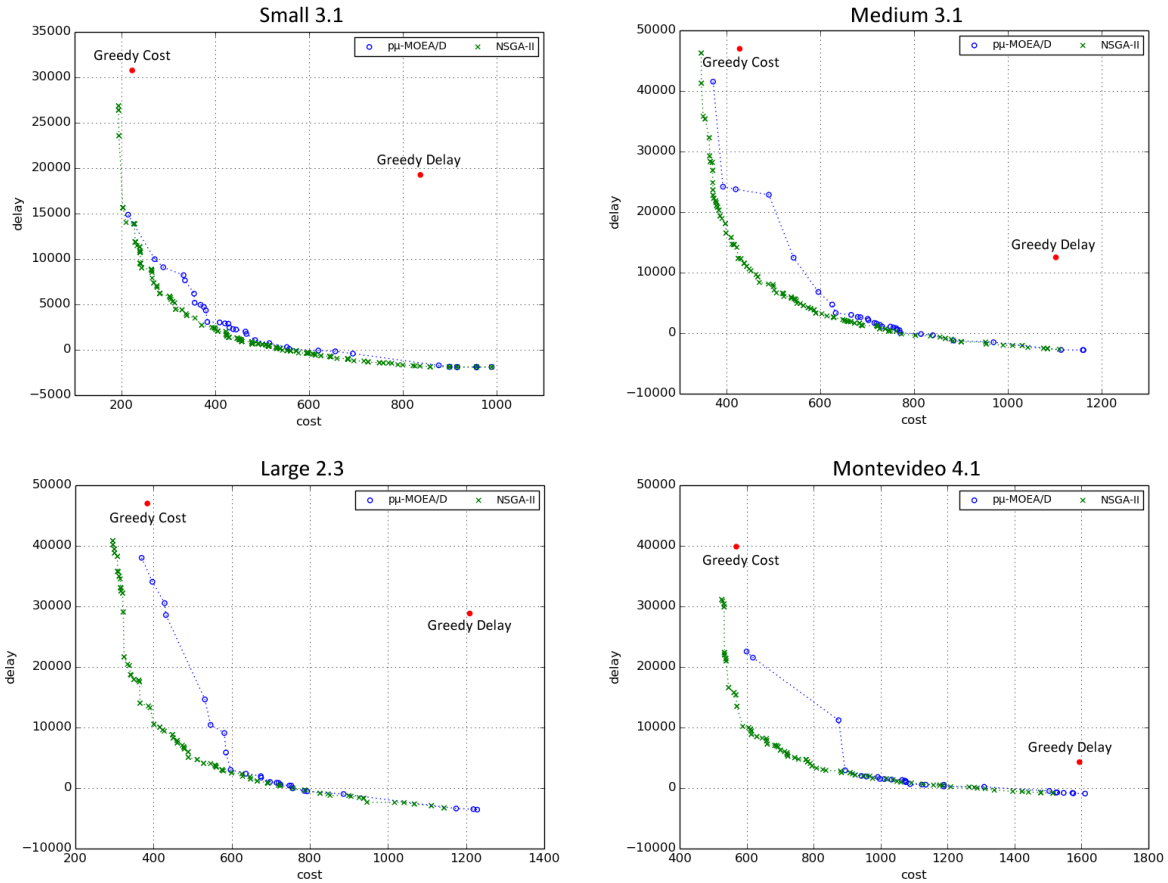
Table 1: Comparison of NSGA-II and pμ-MOEA/D for each problem instance class

instance class	**metric**			**NSGA-II**	**pμ-MOEA/D** [14]
small (9–24 passengers)	improvement over greedy (%)	cost	max.	36.0	36.0
			mean	**21.6**	21.0
			σ	11.7	12.2
		delay	max.	122.4	122.4
			mean	111.1	111.1
			σ	5.0	5.0
	execution time (s)		min.	**1.5**	2.1
			mean	**1.8**	5.1
			σ	0.3	2.8
	hypervolume		max.	$\mathbf{2.0\times10^7}$	1.9×10^7
			mean	$\mathbf{5.6\times10^6}$	5.3×10^6
			σ	6.4×10^6	6.0×10^6
medium (26–37 passengers)	improvement over greedy (%)	cost	max.	**35.4**	32.5
			mean	**21.5**	19.9
			σ	10.1	9.3
		delay	max.	135.9	135.9
			mean	120.2	120.4
			σ	8.8	8.8
	execution time (s)		min.	**2.3**	4.2
			mean	**2.8**	10.6
			σ	0.3	4.1
	hypervolume		max.	$\mathbf{1.2\times10^8}$	1.1×10^8
			mean	$\mathbf{4.9\times10^7}$	4.4×10^7
			σ	3.8×10^7	3.4×10^7
large (42–53 passengers)	improvement over greedy (%)	cost	max.	**69.9**	69.0
			mean	**21.2**	17.2
			σ	18.5	19.5
		delay	max.	123.7	131.3
			mean	113.3	116.0
			σ	4.9	7.1
	execution time (s)		min.	**3.3**	7.2
			mean	**3.7**	16.6
			σ	0.3	6.0
	hypervolume		max.	$\mathbf{5.0\times10^7}$	4.3×10^7
			mean	$\mathbf{3.4\times10^7}$	2.9×10^7
			σ	9.3×10^6	8.5×10^6
Montevideo (9–44 passengers)	improvement over greedy (%)	cost	max.	**35.2**	33.9
			mean	**14.6**	14.4
			σ	8.5	8.9
		delay	max.	123.5	123.5
			mean	**116.4**	116.3
			σ	4.5	4.7
	execution time (s)		min.	**1.5**	1.8
			mean	**2.3**	8.4
			σ	0.7	5.7
	hypervolume		max.	$\mathbf{3.1\times10^7}$	2.5×10^7
			mean	$\mathbf{9.6\times10^6}$	8.2×10^6
			σ	1.0×10^7	8.3×10^6

Table 2: Multiobjective optimization metrics computed for NSGA-II

metric		*small*	*medium*	*large*	*Montevideo*
non-dominated points	min.	80.0	80.0	80.0	80.0
	mean	80.0	80.0	80.0	80.0
	σ	0.0	0.0	0.0	0.0
generational distance	min.	0.0	0.7	0.7	0.0
	mean	0.2	1.1	1.1	0.5
	σ	0.3	0.3	0.2	0.4
spacing	min.	63.9	72.2	114.8	39.0
	mean	155.7	138.9	144.2	69.5
	σ	107.3	42.9	30.6	20.5
spread	min.	0.3	0.2	0.3	0.3
	mean	0.8	0.3	0.4	0.5
	σ	0.2	0.1	0.1	0.2
RHV	max.	1.0	0.99	0.99	1.00
	mean	1.00	0.98	0.98	0.99
	σ	0.005	0.005	0.006	0.012

Figure 3: NSGA-II, pμ-MOEA/D [14], and Greedy results for one problem instance of each class.



the best case (21.2% on average) and a greedy algorithm for delay in up to 135.9% (120.2% on average). The NSGA-II reached better results in shorter execution times than our previous pμ-MOEA/D [14].

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 in the solutions.

Problem instances and detailed results are available at `www.fing.edu.uy/inco/grupos/cecal/hpc/AG-Taxi/`.

References

[1] TaxiFareFinder API. [Online 03-2015] `http://www.taxifarefinder.com`.

[2] T. Bäck, D. Fogel, and Z. Michalewicz, editors. *Handbook of evolutionary computation*. Oxford Univ. Press, 1997.

[3] C. Coello, D. Van Veldhuizen, and G. Lamont. *Evolutionary algorithms for solving multi-objective problems*. Kluwer, New York, 2002.

[4] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. J. Wiley & Sons, Chichester, 2001.

[5] G. Fagúndez, R. Massobrio, and S. Nesmachnow. Resolución en línea del problema de viajes compartidos en taxis usando algoritmos evolutivos. In *Proceedings of the 2014 Latin American Computing Conference*, 2014.

[6] N. Fellows and D. Pitfield. An economic and operational evaluation of urban car-sharing. *Transportation Research Part D: Transport and Environment*, 5(1):1–10, 2000.

[7] E. Ferrari, R. Manzini, A. Pareschi, A. Persona, and A. Regattieri. The car pooling problem: Heuristic algorithms based on savings functions. *Journal of Advanced Transportation*, 37:243–272, 2003.

[8] R. Katzev. Car sharing: A new approach to urban transportation problems. *Analyses of Social Issues and Public Policy*, 3(1):65–86, 2003.

[9] A. Letchford, R. Eglese, and J. Lysgaard. Multistars, partial multistars and the capacitated vehicle routing problem. *Mathematical Programming*, 94(1):21–40, 2002.

[10] Y. Lin, W. Li, F. Qiu, and H. Xu. Research on optimization of vehicle routing problem for ride-sharing taxi. In *8th Int. Conf. on Traffic and Transportation Studies Changsha, China*, 2012.

[11] S. Luke. A java-based evolutionary computation research system. `http://cs.gmu.edu/~eclab/projects/ecj/`. Accessed February 2015.

[12] S. Ma, Y. Zheng, and O. Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *IEEE 29th Int. Conf. on Data Engineering*, pages 410–421, 2013.

[13] R. Massobrio, G. Fagúndez, and S. Nesmachnow. A parallel micro evolutionary algorithm for taxi sharing optimization. In *VII ALIO/EURO Workshop on Applied Combinatorial Optimization*, Montevideo, Uruguay, 2014.

[14] R. Massobrio, G. Fagúndez, and S. Nesmachnow. Planificación multiobjetivo de viajes compartidos en taxis utilizando un micro algoritmo evolutivo paralelo. In *X Congreso Español de Metaheursticas, Algoritmos Evolutivos y Bioinspirados*, 2015.

[15] S. Nesmachnow. Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. *Revista de la Asociación de Ingenieros del Uruguay*, 61:12–15, 2010.

[16] S. Nesmachnow. Metaheuristics as soft computing techniques for efficient optimization. In M. Khosrow-Pour, editor, *Encyclopedia of Information Science and Technology*, pages 1–10. IGI Global, 2014.

[17] C-C. Tao and C-Y Chen. Heuristic algorithms for the dynamic taxipooling problem based on intelligent transportation system technologies. In *4th Int. Conf. on Fuzzy Systems and Knowledge Discovery*, pages 590–595, 2007.

[18] C-L. Xin and W-M. Ma. Scheduling for on-line taxi problem on a real line and competitive algorithms. In *Proc. of the Int. Conf. on Machine Learning and Cybernetics*, pages 3078–3083, 2004.