Stability of Marriage and Vehicular Parking

Daniel Ayala, Ouri Wolfson, Bo Xu,
Bhaskar DasGupta, and Jie Lin

University of Illinois at Chicago

Abstract. The proliferation of mobile devices, location-based services and embedded wireless sensors has given rise to applications that seek to improve the efficiency of the transportation system. In particular, new applications are arising that help travelers find parking in urban settings. They convey the parking slot availability around users on their mobile devices. Nevertheless, while engaged in driving, travelers are better suited being guided to an ideal parking slot, than looking at a map and deciding which open slot to visit. Then the question of how an application should choose this ideal parking slot to guide the user towards it becomes relevant.
Vehicular parking can be viewed as vehicles (players) competing for parking slots (resources with different costs). Based on this competition, we present a game-theoretic framework to analyze parking situations. We introduce and analyze parking slot assignment games and present algorithms that choose parking slots ideally in competitive parking simulations. We also present algorithms for incomplete information contexts and show how these algorithms outperform greedy algorithms in most situations.

Keywords: Stable Marriage, Spatio-temporal resources, Vehicular Parking

1 Introduction

Finding parking can be a major hassle for drivers in some urban environments. The advent of wireless sensors that can be embedded on parking slots has enabled the development of applications that help mobile device users find available parking slots around their locations. A prime example of this type of application is SFPark [1]. It uses sensors embedded in the streets of the city of San Francisco, that can tell if a slot is available. When a user wants to find a parking slot in some area of the city, the application shows a map with marked locations of the open parking slots in the area.

While this type of application is useful for finding the open parking slots around you, it does raise some safety concerns for travelers. The drivers have to shift their focus from the road, to the mobile device they are using. Then they have to look at the map and make a choice about which parking slot to choose from all the available slots that are shown on the map. It would be better (safer) if the app just guided the driver to an exact location where they are most likely to find an open parking slot. Then the question arises, which algorithm should the mobile app use to choose such an *ideal* parking location?

Our main concern in this work is to answer the preceding question. Regardless of the safety concerns stated in the previous paragraph, the question still remains relevant. What is the optimal way of moving towards spatially located resources, to obtain a resource, when there is competition for the resource?

Parking can be viewed as a continuous query submitted by mobile devices to obtain information about spatial resources (parking slots). A mobile user wants to know which is the parking slot to visit in order to minimize various possible utilities like: distance traveled, walking distance to their destination, or monetary price of the parking slot. However, parking is also competitive in nature because after making a choice to visit a particular slot, the success in obtaining that slot will depend on if any other vehicles closer to that slot also made the same choice. This competition for resources (slots) lends itself for modeling this situation in a game-theoretic framework. We then present parking slot assignment games (PSAG) for studying competitive parking situations.

Two categories of PSAG will be considered in this work, complete and incomplete information PSAG. For the complete information PSAG, we relate the problem of finding the Nash equilibrium to the Stable Marriage problem [2]. We show the equivalency of Nash equilibria and Stable Marriage assignments for instances of PSAG.

For the incomplete information PSAG, the model that is most realistic and directly applicable to real-life application of parking slot choice, we present a gravitational approach for choosing parking. The Gravity-based Parking algorithm (GPA) is presented for this model. We also present an adaptation for GPA to road networks and show its merits through simulation.

2 General Setup and Notation

The general setup of the parking problem is as follows:

- There are two types of *objects* as follows.
  - A set of n vehicles $V = \{v_1, v_2, \ldots, v_n\}$.
  - A set of m open parking slots $S = \{s_1, s_2, \ldots, s_m\}$.
- $\mathsf{dist} : (V \cup S) \times S \to \mathbb{R}$ is a distance function. It denotes the distance between a vehicle and a slot, or the distance between two slots.
- $\mathsf{cost} : V \times S \to \mathbb{R}$ is a cost function. It denotes the *cost* of a slot $s_j \in S$ to a vehicle $v_i \in V$. This cost is a general cost. It could include the distance from the vehicle to the slot, $\mathsf{dist}(v_i, s_j)$, the walking distance from s_j to v_i's destination, and/or other utilities that v_i cares about when choosing a slot.
- Each vehicle is assumed to be moving independently of all other vehicles at a fixed velocity. Without loss of generality, we assume that the speeds of all vehicles are the same[1].

[1] Otherwise, we simply need to rescale the distances for each vehicle in our algorithmic strategies.

– A valid *assignment* of vehicles to slots is one where each vehicle is assigned to exactly one slot. It can be defined as a function $g : V \rightarrow S$, where $g(v)$ is the assigned slot for vehicle $v \in V$.[2]
– The *cost of an assignment* g for a player $v \in V$, $C_g(v)$, is defined as $\mathsf{cost}(v, g(v))$ if of all players assigned to slot $g(v)$, v is the closest to it; *i.e.*

$$v = \underset{v' \in V : g(v') = g(v)}{\text{argmin}} \{\mathsf{dist}(v', g(v))\}. \tag{1}$$

Here the argmin function returns the parameter that minimizes the given function. If some other vehicle assigned to $g(v)$ is closer to it than v, then v's cost based on g is $C_g(v) = \mathsf{cost}(v, g(v)) + \alpha$, where α is a *penalty* for not obtaining a parking slot.
– The *total cost of an assignment* g, C_g, is defined as:

$$C_g = \sum_{v \in V} C_g(v) \tag{2}$$

It should be noted that this type of model could be generalized to considering mobile agents (vehicles) that are looking to obtain one of a set of static resources (parking slots) on a map. Besides parking, another application that could conform to this model is one where taxicabs (mobile agents) are competing to obtain clients (static resources) that have a location on a map.

3 Parking Slot Assignment Games

One could define a model in which a centralized authority was in charge of assigning the vehicles to slots. This authority would be looking to minimize some system-wide objectives (optimizing social welfare). In the transportation literature this is usually called a system optimal assignment. In [3], we show how this system optimal assignment can be computed in polynomial time. Even though this centralized model shows good computational properties, it is difficult to justify in real life to distributed mobile users that make their own choices. This is because optimizing social welfare may imply that some travelers will incur a greater cost for the good of others.

We then model parking as a competitive game in which individual, selfish players are competing for the available slots. Any game has three essential components: a set of *players*, a set of possible *strategies* for the players and a payoff function (cost function) [4]. The payoff function determines what is the cost to each player based on a given *strategy profile.* If there are n players in the game then a *strategy profile* is an n-tuple in which the ith coordinate represents the strategy choice of the ith player. It basically represents the choices made by the n players.

[2] Based on this definition, there is a difference between where a vehicle is assigned and where a vehicle parks. If more than one vehicle is assigned to the same slot, then the closest one to it will park there. The others are left without parking. This will always happen when $n > m$.

In our case for the parking problem, we can define the *parking slot assignment game* (PSAG) as follows:

- The set of *players* in PSAG is V (the vehicles).
- The set of available *strategies* to each player is S (the slots).
- The *payoffs* (costs) for each player in this game can be defined by the C_g function introduced in section 2. Let $\mathcal{A} = (s_{v_1}, s_{v_2}, \dots, s_{v_n})$ be the strategy profile chosen by the players, *i.e.* slot s_{v_i} is the chosen slot by vehicle v_i, $1 \leq i \leq n$. Let $g(v_i) = s_{v_i}$, then the cost for any player v_i will be $C_g(v_i)$.
- For this game, the penalty of not finding a parking slot, α, will be defined as a large constant quantity.

4 Nash Equilibrium for PSAG

In this section we introduce the Nash equilibrium for PSAG and establish its relationship with the Stable Marriage problem.

The Nash equilibrium [5] is the standard desired strategy that is used to model the individual choices of players in a game. It defines a situation in which no player can decrease its cost by changing strategy unilaterally. The standard definition of Nash equilibrium translates to the following definition for PSAG:

Definition 1 (Nash Equilibrium for PSAG). *Let $\mathcal{A} = (s_{v_1}, s_{v_2}, \dots, s_{v_n})$ be a strategy profile for the PSAG. Let $\mathcal{A}^*_i = (s_{v_1}, s_{v_2}, \dots, s_{v_{i-1}}, s^*_{v_i}, s_{v_{i+1}}, \dots, s_{v_{n-1}}, s_{v_n})$, for $s^*_{v_i} \neq s_{v_i}$. Let g be the assignment function obtained from strategy profile $\mathcal{A}$ and g^*_i be the assignment function obtained from strategy profile $\mathcal{A}^*_i$. Then strategy profile $\mathcal{A}$ is a Nash equilibrium strategy for the players if $C_g(v_i) \leq C_{g^*_i}(v_i)$ for all i and any $s^*_{v_i} \neq s_{v_i}$.*

$\mathcal{A}^*_i$ is the strategy profile obtained by only player v_i changing strategy from s_{v_i} to any $s^*_{v_i} \neq s_{v_i}$ for any $1 \leq i \leq n$. If the condition in the definition holds then it means that no player can improve by him alone deviating from the Nash equilibrium strategy. For the remainder of the paper, *equilibrium* and *Nash equilibrium* will be used interchangeably.

4.1 Stability of Marriage in PSAG

A vehicle's preference in PSAG is to minimize its cost. Then a vehicle v's preference is to obtain the slot s that minimizes the function $\mathsf{cost}(v, s)$. Then, we say v prefers slot s over slot s' if $\mathsf{cost}(v, s) < \mathsf{cost}(v, s')$. Suppose that the slots had a similar preference order in which a slot s prefers a vehicle v over v' if v is closer to it than v', *i.e.* $\mathsf{dist}(v, s) < \mathsf{dist}(v', s)$.

Definition 2 (Unstable Marriage [2] in PSAG). *An assignment of vehicles to slots is called unstable if there are vehicles v_i and $v_{i'}$, assigned to slots s_j and $s_{j'}$ respectively, but $v_{i'}$ prefers s_j over $s_{j'}$ and s_j prefers $v_{i'}$ over v_i.*

In the following sections, we will show the relationship between the Nash equilibrium for PSAG and stable marriage assignments.

4.2 Computing the Psag Nash Equilibrium

Now we show that we can compute the Nash Equilibrium for Psag by computing stable marriages between the vehicles and the slots.

Theorem 1. *Suppose that the vehicles' preference order is determined by the* cost *function and the slots' preference order is determined by* dist *function. Then an assignment g is a Nash equilibrium if and only if g is a stable marriage between the vehicles and slots.*

Proof. ($\rightarrow$) Let g be an assignment that is a Nash Equilibrium. Then for any $v \in V$, if v deviates strategy unilaterally from $g(v)$, v's cost will increase.

Suppose to the contrary that g is not a stable marriage between vehicles and slots. Then there exist $v, v' \in V$ and $s, s' \in S$ such that $g(v) = s$ and $g(v') = s'$ but v prefers s' over s and s' prefers v over v'. Then the following inequalities hold:

$$\mathsf{cost}(v, s') < \mathsf{cost}(v, s)$$
$$\mathsf{dist}(v, s') < \mathsf{dist}(v', s')$$

But then if v deviates to strategy s' his cost will improve because v is closer to s' than v', and choosing s' has a lesser cost to him than his current choice s. This violates the Nash equilibrium assumption. Contradiction.

($\leftarrow$) Now let g be an assignment that is a stable marriage between vehicles and slots according to their preferences.

Suppose to the contrary that g is not a Nash equilibrium. Then there exists a vehicle that can deviate from the strategy given by g and improve its obtained cost. Let $v \in V$ be such a vehicle and let $g(v) = s$, where $s \in S$. Then v can choose a strategy $s' \neq s$ and improve its obtained cost. Suppose that slot s' was assigned to vehicle v', *i.e.* $g(v') = s'$.

There are two cases to consider.

Case 1: $C_g(v) = \mathsf{cost}(v, s)$

If $C_g(v) = \mathsf{cost}(v, s)$ then by definition v was the closest vehicle to s amongst those that chose s. Now suppose that v can improve its obtained cost by deviating to another strategy s'. If v improves its obtained cost it means that he will obtain his new chosen slot s', otherwise he would pay a penalty α that is larger than his previous cost. Then,

$$\mathsf{dist}(v, s') < \mathsf{dist}(v', s') \tag{3}$$

If v improves its obtained cost then it also means that his obtained cost on the new slot is better than the one he was paying with his previous slot. Then,

$$\mathsf{cost}(v, s') < \mathsf{cost}(v, s) \tag{4}$$

Condition (3) implies that slot s' preferred v over v'. Condition (4) implies that vehicle v preferred s' over s. These two conditions together are a violation of marriage stability. Therefore, g is not a stable marriage. Contradiction.

Case 2: $C_g(v) = \mathsf{cost}(v, s) + \alpha$
This is a case where $n > m$ and v chooses s but does not obtain it. We assume that for vehicles that will not obtain a slot, the stable marriage algorithm will simply assign the vehicle to its smallest cost slot. Suppose v can deviate to s' and improve its cost.

This means that it would definitely obtain the new slot and improve its cost that way (if the slot is not obtained then there's no way of improving). Then,

$$\mathsf{dist}(v, s') < \mathsf{dist}(v', s') \tag{5}$$

For this case, condition (5) is sufficient to show that g is not a stable marriage. v will not obtain any slot according to assignment g, and by (5), s' prefers v over v'. Then v should have been assigned to s' in the first place by the stable marriage algorithm since v really has no partner. Therefore, g is not a stable marriage. Contradiction.

Then by the contradictions obtained in both cases it follows that no vehicle could have improved by deviating strategies from those defined by the assignment g. Therefore, g is a Nash Equilibrium assignment.

By the equivalency obtained between the Nash equilibrium for Psag and stable assignments in Psag, one can compute an equilibrium by finding a stable assignment between the vehicles and slots. Then we found the equilibrium for this two-sided matching problem between agents (vehicles) and spatially located items (parking slots) by assigning preference orders (based on distance to agent) to the items.

5 Gravitational Strategies for Incomplete Information Context

5.1 Incomplete Information Psag

We've shown how one can compute the Nash Equilibrium for Psag by computing a stable marriage assignment between the vehicles and the slots. But this equilibrium is applicable only in a complete information setting. This is one where the vehicles are aware of what their payoffs will be based on their decisions and the decisions of others. For Psag, this means that each vehicle is aware of the locations of all the other vehicles and are aware of their cost functions.

This complete information model is hard to justify in practice because of privacy and security concerns. Not all vehicles will be willing to share the location information at all times. Furthermore, tracking the locations of vehicles at all times, and sharing the locations of all of them with all the users of a system so that they can have up-to-the-second location data on all other potential parking competitors seems infeasible.

Then we wish to analyze PSAG in an incomplete information context. In this context, the players have no knowledge about the locations of the other players. Since they do not have complete access to the distance function, dist, then they have no way of knowing the payoff function for this game; *i.e.* given a strategy profile, none of the players have a way of knowing what its payoff will be.

In the incomplete information PSAG, players make some prior probabilistic assumptions about the locations of the other vehicles in the game and the analysis is performed based on the expectations given by the prior distributions. One can compute the expected costs based on the distribution that is used to denote the location of a vehicle. Then a player will be looking to minimize its expected cost. In this context, the analysis will compute the Nash equilibrium strategies for the players but considering expected costs. This equilibrium is analogous to the Nash equilibrium for PSAG (Definition 1) but instead of using cost given by the cost functions (C_g), it uses expected cost.

For this work, each player will assume that other players are distributed uniformly across the map. Unfortunately, computing the equilibrium for this incomplete information context is very complicated in general, even for simple cases in the number line [3]. Then heuristics are needed to compute ideal strategies for players in this more realistic model.

5.2 Gravity for Parking

The heuristic we want to introduce is one that pushes vehicles towards areas where they are most likely to find a parking slot. Since all other vehicles are assumed to be distributed uniformly across space, this will increase the probability of finding a parking slot upon arrival to the area with a larger amount of available slots. Also, we want the algorithm to take into account the vehicle's location and its proximity to the surrounding slots. In [3], we proposed the Gravity-based Parking Algorithm (GPA), which encompasses these desired properties by using vector addition of force vectors.

In the GPA, slots are said to have a gravitational pull on the vehicles. At any point in time, each slot has a gravitational force on the vehicle that will depend on the distance from the vehicle (magnitude) and location of the slot (direction). So then for each slot, a force vector is generated around the vehicle. Then, all of these vectors are added and the vehicle moves in the direction of the resultant vector (total gravitational force) for a specified time step. Then the process is repeated at the beginning of each time interval.

The classical formula for gravitational force is $F = \frac{Gm_1m_2}{d^2}$ where G is the gravitational constant, m_1 and m_2 are the masses of the respective objects and d is the distance between the objects. But for our purposes we can assume that the masses of the objects are constant. We want to compute the vector that represents total gravitational force generated by all the available slots to a vehicle and use the direction of that vector to move the vehicle in that direction. Then we consider a more simplified formula for gravitational force, since all the masses are constant, represented by:

$$F(v, s) = 1/\mathsf{dist}(v, s)^2 \tag{6}$$

$F(v, s)$ is the gravitational force generated by slot s towards vehicle v.

To consider general costs, this formula can be generalized to:

$$F(v, s) = 1/\mathsf{cost}(v, s)^2 \tag{7}$$

With formula (7), one will compute gravitational pull by considering the general cost as the distance between the vehicle and the slot.

5.3 Gravity-based Parking Algorithm (GPA)

Let z denote the velocity of each vehicle (in units/s), which is constant for all vehicles. Each time step for the algorithm will be 1 second. Each vehicle v will perform the following steps in order to move one time-step at a time towards a parking slot:

– Let S' be the set of currently available slots (updated at every time step). Then for each $s \in S'$ generate vector of magnitude $F(v, s)$ that starts at v's location in direction of s.
– Add the computed force vectors and the result will be the total gravitational force generated by all the available slots on v.
– Move z units (velocity) in the direction given by the total force vector. If the closest slot to v is at a distance less than z then move straight to the closest slot.

These steps define the proposed heuristic for vehicles to use in the incomplete information Psag. The intuition behind the algorithm is that a vehicle is better served moving towards areas of higher density of parking slots when the force to closer slots (determined by distance to them) is not strong enough.

Figure 1 shows what a gravitational force field generated by five sample slots would look like. The arrows represent the direction at which a vehicle will move when it is located at the start point of the arrow and the small dots represent the slots. This diagram gives us an idea of how vehicles move across the map when using GPA and it shows that they will eventually converge to a slot. The GPA was evaluated and performed well in simulations against a greedy approach [3].

6 GPA on a Road Network

On a real-world road network, vehicles are constrained to move only on roads. In this setting we will still use a gravitational approach. It will also be based on the gravity equation defined by equation (6), but now the distance between a vehicle and slot is computed by using the travel distance across the network.

A vehicle can only make a routing choice upon arrival to an intersection, whereas before (in free-space) a vehicle could change direction at any point in time. Therefore, the GPA algorithm will only be used at each intersection by each vehicle. The road network is modeled as a graph $G = (N, E)$ where the vertices (N) represent intersections and the edges (E) are the road segments that connect the intersections.

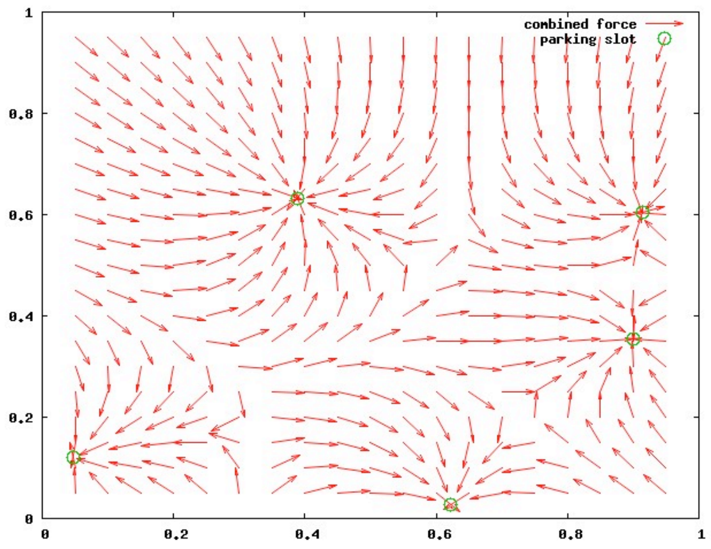

Fig. 1. Force field generated by 5 slots

Instead of adding up all the gravity vectors for all slots (as in Euclidean space), the vehicle will aggregate the gravity information for all slots into special direction vectors (one for each possible direction out of the intersection). Suppose that the intersection where vehicle v is located has k outgoing edges $e_1, e_2, ..., e_k \in E$. Then there will be k direction vectors $g_1, g_2, ..., g_k$ where each vector will have a direction according to its respective embedded edge. The magnitudes of these vectors will start at 0.

Then for each slot s, the shortest path is computed from v to s and the gravity force g is computed using equation (6). Let e_i be the first edge to be taken according to the computed shortest path. Then g_i is updated to be $g_i = g_i + g$.

After repeating this procedure for each slot, the vehicle will use the computed direction vectors $g_1, g_2, ..., g_k$ to make its route choice. From this point, we will introduce two variants of the GPA algorithm that will be evaluated as candidate algorithms for using a gravity-based approach for parking on embedded road networks. The two variants will only differ in how the eventual edge to be taken is computed based on the direction vectors.

6.1 Deterministic Angular GPA (DA-GPA)

In the Deterministic Angular GPA (DA-GPA) the direction vectors $g_1, g_2, ...g_k$ will be added to produce a resultant vector r. This resultant vector will be located between two of the directions to choose from, say $e_i \in E$ and $e_j \in E$. Let θ_i be the angle distance between r and e_i and θ_j be the angle distance between r and

e_j. Then, if $\theta_i < \theta_j$, v will choose e_i as the next edge to travel, otherwise it will choose e_j as the next edge to travel through.

6.2 Randomized Magnitude GPA (RM-GPA)

In the Randomized Magnitude GPA (RM-GPA) the direction vectors $g_1, g_2, ..., g_k$ will be used as part of a probabilistic scheme. Let $T = |g_1| + |g_2| + ... + |g_k|$, *i.e.* the addition of the magnitudes of the k direction vectors. Then let $p_i = |g_i|/T$ for $1 \leq i \leq k$. Then each edge $e_i \in E$ which is an outgoing edge of v's current intersection will be chosen with probability p_i.

6.3 Deterministic Magnitude GPA (DM-GPA)

In the Deterministic Magnitude GPA (DM-GPA) the direction vectors $g_1, g_2, ..., g_k$ will be used to choose the next direction to move towards. The direction with the vector with the largest magnitude will be chosen.

The efficiency of these three methods will be evaluated through simulation.

7 Simulation and Results

In this section we will evaluate DA-GPA, RM-GPA, and DM-GPA against the greedy parking algorithm. The greedy algorithm simply moves each vehicle towards its current closest slot.

7.1 Simulation Environment

The simulation tests the three GPA variants with varying number of values of n and m for the embedded road network in Euclidean space. The simulation is run on a one mile by one mile map where roads are generated that either run from east to west or north to south. Locations for slots on these roads are pre-generated as well.

The map is then partitioned into 16 equal-sized square regions. A random permutation of the regions is generated (uniform distribution) and is used as the ranking of the popularity of each region for available slots. To choose each of the m open slots, first a random number is generated to determine which region to choose a slot from. The Zipf distribution with its skew parameter and the regional popularity previously generated are used to generate this random number. Then a random slot (uniform) is chosen from the region denoted by the Zipf number. The n vehicles' initial positions are generated using the uniform distribution on the grid.

After generating the vehicles and slots, the algorithms are tested. The GPA algorithms were tested against the greedy parking algorithm, which simply moves each vehicle towards its current closest slot. For the GPA algorithms, the vehicles will move as described in the procedures on section 6.

When a vehicle reaches an open parking slot, the time it took for it to find that slot is saved. Then a new slot is chosen randomly (uniform) on a randomly chosen region (Zipf). Also a new vehicle is generated at a random location on the grid (uniform). The simulation run stops when a given time horizon of 3,600 seconds is surpassed.

The parameters for the simulation are:

- n - the number of vehicles.
- m - the number of slots.
- k - the regional skew of the Zipf distribution.

The values that were tested for each parameter are detailed in table 1. For each configuration of the parameters, 20 different simulation runs were generated and tested.

Parameter	**Symbol**	**Range**
Vehicles	n	{40,80}
Slots	m	{20,30,40}
Zipf Skew	k	{0, 1, 2, 3}

Table 1. Parameters tested on Simulation

7.2 Simulation Results

Figure 2 shows the improvement of the DM-GPA algorithm over the greedy parking algorithm. In the best case, the highest improvement that was attained was one of 40% ($n = 40, m = 40, skew = 1$). We can see that the lowest improvement is seen when the skew is 0 (uniformly distributed). Higher improvements are seen in highly skewed situations (skew of 1 or above), although as the regional skew increases past 1, the performance decreases. The RM-GPA and DA-GPA also showed positive improvements over the greedy algorithm but were not better in performance than the DM-GPA. The results for RM-GPA and DA-GPA are thus omitted for space considerations.

8 Conclusions

In this paper our main goal was to analyze vehicular parking. We presented two models that can be used to study the parking problem in a game-theoretic framework.

For the complete information model, in which vehicles are aware of the location and cost information of other players, we presented an algorithm for computing the Nash equilibrium for parking slot assignment games (Psag). We established the relationship between the parking problem and the stable marriage problem. We also showed that the Nash equilibrium was actually equivalent to a stable marriage between vehicles and slots.

For the incomplete information model, vehicles are not aware of the locations of the other mobile users that are also looking for parking. For this model we presented the Gravity-based Parking Algorithm (GPA). For the adapted GPA to road networks we presented the DA-GPA, RM-GPA and DM-GPA. The merits of the GPA's were tested using simulations.

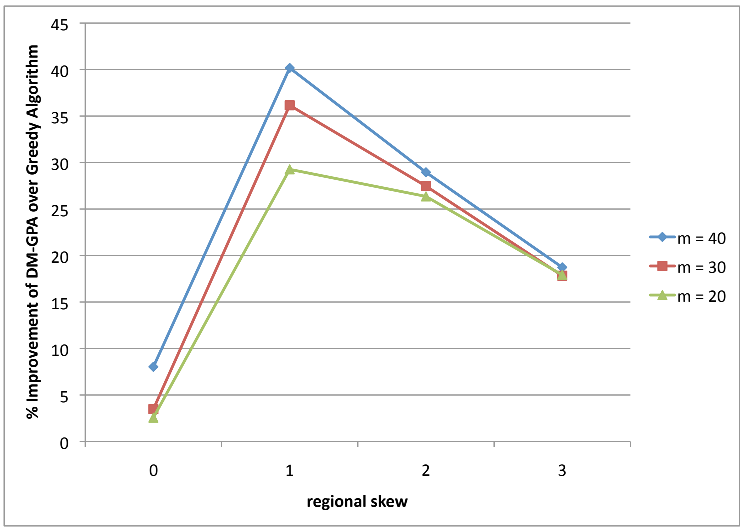

Fig. 2. % improvement of DM-GPA over greedy algorithm ($n = 40$, varying m)

References

1. http://sfpark.org/
2. Gale, D., Shapley, L.: College admissions and the stability of marriage. The American Mathematical Monthly **69**(1) (1962) 9–15
3. Ayala, D., Wolfson, O., Xu, B., Dasgupta, B., Lin, J.: Parking slot assignment games. In: Proc. of the 19th Intl. Conf. on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2011), Chicago, IL (November 2011)
4. Rasmusen, E.: Games and Information. 4th edn. Blackwell Publishing (2006)
5. Nash, J.: Equilibrium points in n-person games. Proceedings of the National Academy of Sciences **36**(1) (1950) 48–49