Parking Slot Assignment Games

Daniel Ayala, Ouri Wolfson, Bo Xu, Bhaskar Dasgupta
University of Illinois at Chicago
Department of Computer Science
851 S. Morgan Street
Chicago, IL 60607, USA
dayala@uic.edu, {wolfson, boxu, dasgupta}@cs.uic.edu

Jie Lin
University of Illinois at Chicago
Department of Civil and Materials Engineering
842 W. Taylor Street (MC246)
Chicago, IL 60607, USA
janelin@uic.edu

ABSTRACT

With the proliferation of location-based services, mobile devices, and embedded wireless sensors, more and more applications are being developed to improve the efficiency of the transportation system. In particular, new applications are arising to help vehicles locate open parking spaces. Nevertheless, while engaged in driving, travelers are better suited being guided to a particular and ideal parking slot, than looking at a map and choosing which spot to go to. Then the question of how an application should choose this ideal parking spot becomes relevant.

Vehicular parking can be viewed as vehicles (players) computing 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 (PSAG) in complete and incomplete information contexts. For both models we present algorithms for individual players to choose parking spaces ideally. To evaluate the more realistic incomplete information PSAG, simulations were performed to test the performance of various proposed algorithms.

1. INTRODUCTION

Finding parking can be a major hassle for drivers in some urban environments. For example in [17], studies conducted in 11 major cities revealed that the average time to search for curbside parking was 8.1 minutes and cruising for these parking spaces accounted for 30% of the traffic congestion in those cities on average. Even if the average time to find parking was smaller, it would still account for a large amount of traffic. Suppose that the average time to find parking were 3 minutes (as opposed to 8.1), each parking space would still generate 1,825 vehicle miles traveled (VMT) per year [18]. That number would of course be multiplied by the number of parking spaces in the city. For example, in a city like Chicago with over 35,000 curbside parking spots [20], the total number of VMT becomes 63 million VMT per year due to cruising while searching for parking. Furthermore, this would account for waste of over 3.1 million gallons of gasoline and over 48,000 tons of CO_2 emissions.

With the advent of location-based services and embedded wireless sensors, applications that enable mobile devices to find open parking spots in urban environments are being developed. A prime example of this type of application is SFPark [1]. It uses wireless sensors embedded in the streets and parking garages of the city of San Francisco, that can tell if a parking spot has opened up. When a user wants to find a parking spot in some area of the city, the application shows a map with marked locations of the open parking spots in the area.

While this type of application may be useful for finding the open parking spots 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 space to choose from all the available spots that are shown in 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 spot. Then the question arises, which algorithm should the mobile app use to choose such an *ideal* parking location?

In this paper, our main concern is to answer the preceding question. To that end, we study the parking problem in various contexts. The first model we study is a centralized model in which some centralized authority makes parking choices for travelers and assigns each to a specific parking slot. We then study a model with distributed selfish agents that are competing for the parking slots. This competition for resources (slots) lends itself for modeling this situation in a game-theoretic framework. We then introduce parking slot assignment games (PSAG) for studying this distributed model.

Two categories of PSAG will be considered, complete and incomplete information PSAG. For the complete information game, an algorithm for computing the Nash equilibrium is presented. Also, we show that the *price of anarchy* (defined in section 4.3 as the ratio between Nash Equilibrium and System Optimal) for the vehicular parking problem is unbounded. For the incomplete information game, the model that is most realistic and directly applicable to real-life applications of parking slot choice, examples are shown to compute the Nash Equilibrium with expected costs and the Gravity-based Parking algorithm (GPA) is introduced. The algorithm is evaluated through simulations. In various cases the algorithm shows an improvement of over 25% compared to the Naïve parking algorithm. This improvement amounts to savings of up to 785,000 gallons of gasoline and 12,000 tons of CO_2 emissions per year in a major city like Chicago.

The rest of the paper is organized as follows. In section 2 we present the general setup of the problem and some notations to be followed in the rest of the paper. In section 3 we present the centralized model and present an algorithm for computing the welfare-optimizing parking assignment. In section 4 we define the Parking Slot Assignment game and prove various properties of the game for the complete information model. In section 5 we introduce PSAG with incomplete information, look at some simple examples that help us define the Gravity-based Parking algorithm. In section 6 we present results and evaluation of simulations that were ran to test the performance of the Gravity-based algorithm over the Naïve parking algorithm. In section 7 we discuss some related work and in section 8 we present some concluding remarks.

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.
- 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].
- 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 $g : V \to S$, 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 $\mathsf{dist}(v, g(v))$ if of all players assigned to slot $g(v)$, v is the closest to it; *i.e.*

$$v = \operatorname*{argmin}_{v' \in V : g(v') = g(v)} \{\mathsf{dist}(v', g(v))\}. \qquad (1)$$

  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 is $\mathsf{dist}(v, g(v)) + \alpha$ where α is a *penalty* for not obtaining a parking slot.
- The *cost of an assignment* g, C_g, is defined as:

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

[1]Otherwise, we simply need to rescale the distances for each vehicle in our algorithmic strategies to be described subsequently.
[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 $n > m$.

3. CENTRALIZED MODEL - OPTIMIZING SOCIAL WELFARE

In this model, a centralized authority is in charge of assigning the vehicles to slots. The authority would be looking to minimize some system-wide objectives. The most common objective function used for optimization is Eq (2), where an assignment g that minimizes the total system cost is computed. This computation is also commonly referred to as one that optimizes the social welfare.

Suppose that $y_{i,j} \in \{0, 1\}$ is the usual indicator variable denoting if $v_i \in V$ is assigned to $s_j \in S$, *i.e.* $g(v_i) = s_j$; then we need to *minimize* $\sum_{i=1}^{n} \sum_{j=1}^{m} y_{i,j}\, \mathsf{dist}(v_i, s_j)$.

Theorem 1. *A system optimal solution can be computed in (strongly) polynomial time.*

Proof. We reduce our problem to an instance of the minimum-cost network flow problem on a directed bipartite graph, for which a strongly polynomial time exact solution is well-known (*e.g.*, see [4]). Since the capacities of each slot in our instance are all integral (equal to 1), the solution to the flow problem will also be integral (*e.g.*, see [13]). We have a vertex v_i for every vehicle v_i, a vertex s_j for every parking slot s_j and a directed edge (v_i, s_j) of weight $\mathsf{dist}(v_i, s_j)$. We also have two additional vertices, a source vertex s and a sink vertex t, with edges (s, v_i) for every $1 \le i \le n$ and edges (s_j, t) for every $1 \le j \le m$ (each of zero weight). Let $G = (V, E)$ be the resulting directed graph. Figure 1 shows what the structure of G will look like. Then, defining the flow by f, we use the following minimum-cost network flow problem on this graph with $min(n, m)$ being the flow requirement:

$$\begin{aligned}
&\textit{minimize} \sum_{\{v_i, s_j\} \in E} \mathsf{dist}(v_i, s_j)\, f(v_i, s_j)\\
&\textit{subject to}\\
&\qquad \forall v_i : 0 \le f(s, v_i) \le 1\\
&\qquad \forall \{v_i, s_j\} \in E : 0 \le f(v_i, s_j) \le 1\\
&\qquad \forall s_j : 0 \le f(s_j, t) \le 1\\
&\qquad \textstyle\sum_{i=1}^{n} f(s, v_i) = \sum_{j=1}^{m} f(s_j, t) = min(n, m)
\end{aligned}$$

The first three constraints in the mathematical program above are capacity constraints and the final one is the flow requirement constraint. Finally, if $f(v_i, s_j) = 1$ then v_i is assigned to s_j, $g(v_i) = s_j$.

If n > m, then some vehicles will remain unassigned. These unassigned vehicles can be assigned to any slot. They will not obtain any slot they are assigned. Suppose that one of these unassigned vehicles could obtain some slot s, then the minimum assignment would be one in which that vehicle would be assigned to s and the original vehicle assigned to s would be assigned to any other slot. This assignment would have a smaller cost than the one computed by the program above, which would be a contradiction. $\square$

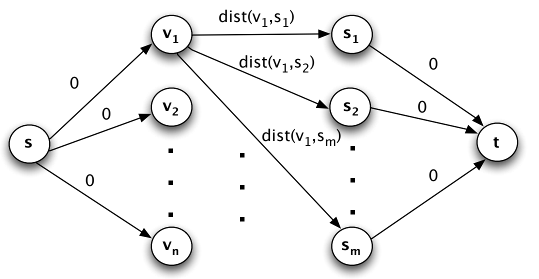

Figure 1: Graph Construction for Network Flow Problem with weights (all capacities are 1)

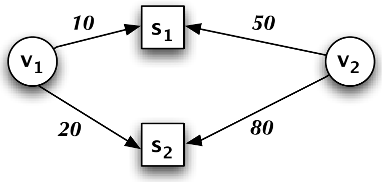

Figure 2: Optimizing Social Welfare Example

Even though this model shows good computational properties, it may be difficult to justify in real life to distributed users that make their own choices. This is because optimizing social welfare implies that some travelers may incur a greater cost for the good of others.

For example, in the parking problem shown in Figure 2, the assignment that optimizes social welfare is one such that $g(v_1) = s_2$ and $g(v_2) = s_1$. This assignment has a cost of 70. The other possible assignment has a higher cost of 90, but it is more profitable for v_1 since he would cut his cost in half. Then v_1 has incentive to deviate from the optimal assignment and choose s_1. This then leads us to analyze the problem with distributed agents where each traveler makes parking choices that benefit himself.

4. DISTRIBUTED MODEL WITH COMPLETE INFORMATION

In this model, there is no central authority and each vehicle is an independent agent that makes its own parking slot choice. This version of the problem can be analyzed as a game where vehicles acting as players are competing for parking slots and they wish to reduce the cost that is obtained based on their choices and the choices of the other players.

4.1 PSAG with Complete Information Definition

Any *game* has three essential components: a set of *players*, a set of possible *strategies* or choices for the players, and a *payoff function* (cost function) [15]. 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.

In this complete information model, all vehicles have access to all of the information about the locations of the other vehicles. In other words, they know the payoff function for this problem.

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 $\mathcal{A} = (s_{v_1}, s_{v_2}, \ldots, s_{v_n})$ 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.

The Nash equilibrium [10] 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 2 (NASH EQUILIBRIUM FOR PSAG). *Let $\mathcal{A} = (s_{v_1}, s_{v_2}, \ldots, s_{v_n})$ be a strategy profile for the* PSAG. *Let $\mathcal{A}_i^* = (s_{v_1}, s_{v_2}, \ldots, s_{v_{i-1}}, s_{v_i}^*, s_{v_{i+1}}, \ldots, s_{v_{n-1}}, s_{v_n})$, 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 an 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.

4.2 An Equilibrium Strategy for PSAG

Algorithm 1 computes the assignment function g that leads to an equilibrium strategy for each player. It does so through an iterative process. On each iteration, it chooses the unassigned vehicle (v) that is closest to any of the unassigned slots (s) and sets $g(v) = s$.

The algorithm enters the **if** statement at the end only if the number of vehicles (n) is greater than the number of available slots (m). The vehicles that make a strategy choice in this section are guaranteed to not obtain a parking slot because all of the slots were chosen in the preceding **while** loop by vehicles that are closer.

THEOREM 3. *Given a set of vehicles V, a set of open parking slots S and a distance function* dist *that defines the distances of each vehicle to each slot, then Algorithm 1 computes the assignment g that leads to a Nash equilibrium strategy for each vehicle.*

Algorithm 1 Equilibrium Strategy Computation

```
$V' \leftarrow V$
$S' \leftarrow S$
**while** ($V' \neq \emptyset$) AND ($S' \neq \emptyset$) **do**
  $v, s \leftarrow \underset{v_i \in V', s_j \in S'}{\operatorname{argmin}} \{\mathsf{dist}(v_i, s_j)\}$
  $g(v) \leftarrow s$
  $V' \leftarrow V' \setminus \{v\}$
  $S' \leftarrow S' \setminus \{s\}$
**end while**
**if** $V' \neq \emptyset$ **then**
  **for all** $v \in V'$ **do**
    $g(v) \leftarrow \underset{s \in S}{\operatorname{argmin}}\{\mathsf{dist}(v, s)\}$
  **end for**
**end if**
```

PROOF. Let g be the assignment that is computed by algorithm 1. There are two cases to consider as to where the assignment of a slot for player v_i took place. Either v_i was assigned its slot in the **while** loop or in the **if** statement of the computation.

Suppose v_i was assigned its slot in the **while** loop. Then that means that at that point of assignment he is the closest to any of the slots that have remained unassigned. Deviating strategies to any other unassigned slot does not improve his cost because it is a larger cost than the one he would obtain by choosing $g(v_i)$ (by usage of argmin). Deviating strategies to any one of the already assigned slots will not improve his cost because the players assigned to those slots were closer to them than him so he would not obtain it and would have to pay a larger cost because of the penalty α for not obtaining the slot. Then, in this case no player assigned a slot in the **while** loop has incentive to deviate to another strategy.

Now suppose v_i is assigned its slot in the **if** statement of the computation. Then no matter what slot he chooses, he will not get it because they are already assigned to players that are closer to all of them than he is. Then by default, choosing his closest parking spot is his equilibrium strategy. Therefore by using Algorithm 1 to compute g, no player v_i has an incentive to deviate from strategy $g(v_i)$ and so g defines an equilibrium assignment. □

Then for this version of the problem, we can compute the equilibrium in a simple way. The problem is that it suffers from security and privacy concerns. Not all travelers are willing to share location information or are comfortable with being tracked at all times. Therefore, we will analyze in section 5 this same problem but with incomplete information so that users would not have the necessity of sharing their locations at all times.

4.3 Price of Anarchy

The *price of anarchy* (POA) of a game is the ratio of the total cost paid in the equilibrium assignment over the total cost paid by the players in the assignment that minimizes the social welfare [11]. The POA of the PSAG class of games will be the largest such ratio that could be found for an instance of the PSAG. In this section we show that the POA of the PSAG with complete information is unbounded. This is a fundamental result which indicates that for the vehicular parking problem, when travelers are selfish and make their own choices, costs can get arbitrarily worse than the optimal assignment. More specifically, the ratio between the costs of the equilibrium and optimal parking assignments can grow unboundedly as the size of the problem increases.

THEOREM 4. *The Price of Anarchy in the Complete Information* PSAG *is unbounded.*

PROOF. Consider a game with n vehicles and n open parking slots. Let $A = \{a_{ij}\}$ form a matrix, where the entry $a_{ij} = \mathsf{dist}(v_i, s_j)$ is the distance from v_i's original position and s_j.

Let the distance function be defined by the following matrix:

$$\{d(v_i, s_j)\} = \begin{bmatrix} n & 2n & \cdots & n \cdot n \\ n^2 & 2n^2 & \cdots & n \cdot n^2 \\ \vdots & \vdots & \ddots & \vdots \\ n^n & 2n^n & \cdots & n \cdot n^n \end{bmatrix}$$

Then $\mathsf{dist}(v_i, s_j) = jn^i$. Any assignment in this game will lead to a total cost (adding the costs of all players) of some polynomial of degree at least n.

The equilibrium assignment will be one that is obtained by performing the Algorithm 1. In this method, the smallest entry, a_{ij}, in the matrix will be computed and v_i will be assigned to parking s_j. Then that column and row will be removed from the matrix and the smallest entry will be computed in this updated matrix. It's clear that the polynomial that is obtained from this algorithm will be the one that adds up the diagonal entries of matrix A. Then the equilibrium assignment will have a total cost of:

$$EQ = \sum_{k=0}^{n-1} (n-k)n^{n-k} = n^{n+1} + \sum_{k=1}^{n-1} (n-k)n^{n-k} \qquad (3)$$

Then, the equilibrium solution leads to a total cost that is given by a polynomial of degree $n + 1$.

Now consider an assignment that tries to choose the smallest distance for the larger cost vehicles. So then for v_n, slot s_1 will be assigned so as to minimize the large cost factor that it has of n^n. This assignment chooses the diagonal that starts on position a_{n1} and ends in position a_{1n}. For this special assignment, v_i is assigned slot s_{n-i+1}. Let the cost of this assignment be defined as X. Then:

$$X = \sum_{k=0}^{n-1} (k+1)n^{n-k} = n^n + \sum_{k=1}^{n-1} (k+1)n^{n-k} \qquad (4)$$

If we take the ratio of EQ/X then we get a polynomial of degree 1 which is a function of n. Let OPT be the cost of the assignment that optimizes the social welfare. By definition $OPT \leq X$. Therefore $EQ/X \leq EQ/OPT$ and then EQ/OPT will be greater than some function of n. Therefore, the POA given by the EQ/OPT ratio will be unbounded because an instance of PSAG can be constructed by the previous construction that will be larger than any proposed bound. □

The price of anarchy is also used for computing bounded approximations to problems in a distributed manner when the

POA is bounded (*e.g.* [8, 11]). When the POA is bounded, individual agents using Nash equilibrium strategies compute an approximation for the optimal problem with the POA being the approximation ratio. The fact that the POA is unbounded makes this technique impossible for this problem.

5. DISTRIBUTED MODEL WITH INCOMPLETE INFORMATION

In this model, each vehicle is again an independent agent that makes its own parking slot choice. It will be analyzed as an *incomplete information game*. It is considered an incomplete information game because the players have no knowledge about the other players' distances to the slots. 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.

5.1 Model Changes

5.1.1 Locations of Other Players

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 2) but instead of using cost given by the cost functions (C_g), it uses expected cost.

5.1.2 Updated Information

In the previous models, having updated information about parking slots no longer being available was irrelevant. In the centralized case the centralized authority chose a slot for each vehicle so updated information would not change a choice for a vehicle. In the complete information model, each vehicle executing an equilibrium strategy knew which would be his equilibrium choice and having updates on parking spaces being taken would not change those.

On the contrary, in the incomplete information model this property changes because the analysis is performed in expectation. The analysis for a game with n players where you don't know where the other $n-1$ players are located is different than the analysis for a game with $n-1$ players where you don't know where the other $n-2$ players are located. Also, a vehicle will make a choice of a slot based on the m slots that are available, if one of those is no longer available then the probabilistic analysis will be different when the situation becomes that now there are $m-1$ available slots.

Then, for the incomplete information case, we assume that players receive updates about slots that are no longer available. Then a game for this model analyzes the situation until only one vehicle finds a parking spot instead of analyzing how all vehicles are going to find or not find parking slots like in the other models. Basically, when an update of a slot no longer being available is received, a new game is analyzed with the updated parameters and an updated strategy is computed.

5.1.3 Expected Cost Function

For this version of the PSAG, the penalty of not finding a parking spot α is also different. Since the game only analyzes the situation until one vehicle finds an open slot, then the cost should be designed so that a vehicle is not only more likely to find a parking slot but also is better positioned for subsequent iterations of the game. Then the expected cost in this game is the expected distance traveled by the player plus the penalty α (if the player did not find the open parking space). α is defined as the expected average distance to the closest $n-1$ available parking spots after traveling the distance to its new expected updated location.

5.2 Simple Examples

5.2.1 2 vehicles, 3 slots on the Line

In this section we consider an example of the incomplete information PSAG played in the [0,1] line. Let $n = 2$ and let one player be named v_x and the other named v_y. Let $x \in [0,1]$ and $y \in [0,1]$ be the locations of the players respectively. Let $m = 3$, where the location of s_1 is 0 (left end of line) and the location of s_2 and s_3 is 1 (right end). In this model, each player does not know the location of the other and assumes that its location is distributed uniformly in $[0,1]$.

We'll assume that the strategies can be randomized (mixed) and that they will depend on the players' locations. In randomized strategies players choose each strategy with a probability. Let $p : [0,1] \to [0,1]$ be a function that maps the location of a player to a probability. In other words, $p(x)$ defines the probability that v_x chooses to move to the left to find a parking slot and $1 - p(x)$ will be the probability of him moving to the right. Since the game is symmetric, in terms of the players, we assume that the function will be the same for each player.

Now suppose that $x \leq 1/2$. Then the expected cost for v_x can be defined, based on definition of expectation, as:

$$\begin{aligned} C^*(x) &= \int_0^x \{p(y)[p(x)(1-x+2y) + (1-p(x))(1-x)] \\ &\quad +(1-p(y))[p(x)(x) + (1-p(x))(1-x)]\}dy \\ &+ \int_x^{\frac{1}{2}+x} \{p(y)[p(x)(x) + (1-p(x))(1-x)] \\ &\quad +(1-p(y))[p(x)(x) + (1-p(x))(1-x)]\}dy \\ &+ \int_{\frac{1}{2}+x}^1 \{p(y)[p(x)(x) + (1-p(x))(1-x)] \\ &\quad +(1-p(y))[p(x)(x) + (1-p(x))(2+x-2y)]\}dy \end{aligned}$$

Now let $p^*(x)$ be defined as follows:

$$p^*(x) = \begin{cases} 1 & \text{if } x \leq 3/8 \\ 0 & \text{if } x > 3/8 \end{cases} \qquad (5)$$

THEOREM 5. *In an incomplete information* PSAG *played* *on the [0,1] line with $n = 2$, $m = 3$, with the location of the slots at 0,1, and 1; the strategy $p^*(x)$ is a Nash equilibrium.*

Proof. Let $n = 2$ with players named v_x and v_y with $x \in [0, 1]$ and $y \in [0, 1]$ being their positions respectively. Also let $m = 3$ with the positions of each slot being 0,1, and 1.

Suppose that $x < 3/8$ and suppose that v_y uses strategy p^*. We prove the theorem by showing that the smallest expected cost for v_x is attained by using the same strategy. Based on v_y's strategy choice, the expected cost becomes:

$$\begin{aligned} C^*(x) &= \int_0^x [p(x)(1-x+2y) + (1-p(x))(1-x)]dy \\ &+ \int_x^{3/8} [p(x)(x) + (1-p(x))(1-x)]dy \\ &+ \int_{3/8}^{\frac{1}{2}+x} [p(x)(x) + (1-p(x))(1-x)]dy \\ &+ \int_{\frac{1}{2}+x}^{1} [p(x)(x) + (1-p(x))(2+x-2y)]dy \end{aligned}$$

Now after simplifying the updated equation we obtain a linear function in terms of $p(x)$. Then the expected cost equation becomes of the form $C^*(x) = ap(x) + b$. In this case $a = 2x - \frac{3}{4}$. When $a < 0$, $p(x) = 1$ is optimal for v_x and when $a > 0$ then $p(x) = 0$ is optimal for v_x.

Then solving for x when $a < 0$ gives us that $p(x) = 1$ when $x < 3/8$ and $p(x) = 0$ when $x > 3/8$, in which case $p(x) = p^*(x)$. Thus it turns out that randomized strategies are not needed. Based on the player's location, he'll know which strategy to choose.

The same result holds when $x > 3/8$. $\square$

5.2.2 2 vehicles, 2 slots on the Line

If we take the same example as in the previous section but remove one of the slots on the right end of the line then the following theorem applies (proof not shown due to space constraints):

Theorem 6. *In an incomplete information* Psag *played on the [0,1] line with $n = 2$, $m = 2$, with the location of the slots at 0 and 1; the strategy where $p(x) = 1$ if $x \leq 1/2$ and $p(x) = 0$ if $x > 1/2$ is a Nash equilibrium.*

5.3 Gravitational Model for Parking

Solving the incomplete information Psag for arbitrary values of n and m is difficult in general because of the different combinations of strategies to consider from all players when constructing the expected cost formula. The general number of slots also increases the number of strategies for each player and further complicates the expected cost formula. We then wish to propose a heuristic based on the results of section 5.2 with which vehicles can compute their strategies in an incomplete information context.

The heuristic we will introduce is based on a gravitational force model. Another technique that could be suitable for this problem is greedy algorithms. We will test our gravitational algorithm against a greedy approach that chooses to move towards the closest slot at all times. An approach that models the problem as a Traveling Salesman problem [22] would not be suitable for our model since we assume that updated information is available at all times so choosing to visit one first slot is as good as planning a tour of the slots.

5.3.1 Gravity Force for Equilibrium Parking

From the examples on the line we know that the equilibrium strategies are ones in which a player should always choose a parking slot (no randomization) depending on his location on the map. For the example in section 5.2.2 the player should always choose s_1 (move left) if he's located in $[0, 1/2]$ and move right otherwise. For the problem in section 5.2.1 he will move left when located in $[0, 3/8]$ and move right otherwise. So then the fact that there are two available parking slots on the right changes the bounds at which it will be more profitable for the player to choose to move left or right.

This observed phenomenon can be modeled as parking slots having some type of gravitational pull on the vehicles. In physics, *gravitational force* is determined by the masses of the objects (slots and vehicles) and the distance between them. If we set the masses of all slots and vehicles to be constant then it is expected that in the $n = 2$, $m = 2$ problem in section 5.2.2 the point where the forces are equal would be right in the middle of the line segment (distances are the same). This point changes when adding the third available slot because the two slots to the right will have more gravitational pull than the lone slot in the left. That is why it is an equilibrium for a player to move to the right whenever it is located anywhere in $(3/8, 1]$.

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. The exponent of 2 in the denominator is a parameter of the space being considered. 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:

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

$F(v, s)$ is the gravitational force generated by slot s towards vehicle v and β is a parameter to be determined experimentally because it may vary for different instances of the Psag problem. This optimal value for β will change for different values of n and m, and for differing geographical locations of the slots.

5.3.2 Gravity-based Parking Algorithm

Let z denote the velocity of each vehicle (units/s) (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. The merits of the algorithm will be determined experimentally through simulations.

6. SIMULATIONS AND RESULTS EVALUATION

6.1 Simulation Framework

Simulations were performed to evaluate the performance of the Gravity-based Parking Algorithm (GPA). They were implemented in the Java language version 1.6.

6.1.1 Simulation Setup

The simulation tests the GPA with varying number of values of n and m for the 2-dimensional Euclidean space in the unit square. The unit square is first 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. Then the number of parking slots per region is determined by using the Zipf distribution based on the ranking of the region and the skew parameter used for the Zipf distribution. Then for each of the m slots a Zipf number between 1 and 16 is generated to determine its region, then inside that region its position is determined using the uniform distribution.

The n vehicles' positions are generated using the uniform distribution on the unit square.

After generating the vehicles and slots, the algorithms are tested. The GPA is tested against the Naïve Parking Algorithm, which just makes vehicles always move towards the closest available slot.

For the GPA, vehicles move in a step-by-step fashion as dictated by the steps delineated in section 5.3.2. The simulation is run a second at a time so that vehicles can recompute their total force vectors at each second.

When a vehicle reaches an open parking slot, the distance traveled to the slot is saved. Then a new slot is generated on a randomly chosen (Zipf distribution) region. Also a new vehicle is generated at a random location (uniform distribution). The simulation run stops when a given time horizon 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
- β - The exponent of the force equation (6).
- z - velocity of all vehicles (units/s)
- hmt - The hybrid mode threshold. When the magnitude of the total force vector is less than this threshold then the algorithm uses the Naïve Parking algorithm and moves for that second in the direction of the closest slot. Mainly implemented to guard against the rare situation where all forces cancel out and the vehicle would be headed in no direction based on gravity.

The values that were tested for each parameter are detailed in table 1. For each configuration of the parameters, 1000 different simulation runs were generated and tested.

Parameter	Symbol	Range
Vehicles	n	{40,80}
Slots	m	{20,30,40} when $n = 40$ {40,60,80} when $n = 80$
Zipf Skew	k	{0,1,2}
Gravity Exponent	β	{1,2,3,4,5,6}
Velocity	z	0.01 units/s
Hydrid Thres.	hmt	0.1

Table 1: Parameters tested on Simulation

6.1.2 Comparison Algorithm and Performance Measure

The GPA was tested against the Naïve Parking algorithm (NPA) which simply moves each vehicle towards the closest slot available.

The performance measure to be used in comparing the algorithms is the average distance traveled per vehicle. Since for different values of n or m the value for average distance traveled may be different, then the metric to be analyzed will be *percent improvement* of the GPA over the NPA.

We can use the results of section 5.2.1 to gauge what magnitude of improvement can be expected on this performance metric based on this example. We can compute this analytically for this 3 slot example. If a vehicle were to use the Naïve algorithm in this problem his average expected distance traveled will be:

$$\int_0^{1/2}\left[\int_0^x (1-x+2y)dy + \int_x^1 xdy\right]dx \tag{7}$$

$$+\int_{1/2}^1\int_0^1 (1-x)dydx = \frac{1}{3} \approx 0.333 \tag{8}$$

When a player uses the Nash equilibrium strategy in this problem, his average expected distance traveled will be:

$$\int_0^{3/8}\left[\int_0^x (1-x+2y)dy + \int_x^1 xdy\right]dx \tag{9}$$

$$+\int_{3/8}^1\int_0^1 (1-x)dydx = \frac{163}{512} \approx 0.318 \tag{10}$$

This gives the Nash equilibrium an expected percent improvement of approximately 4.5% over the NPA for this specific problem. Then the improvement of using the Nash equilibrium for this problem is potentially small. Based on this

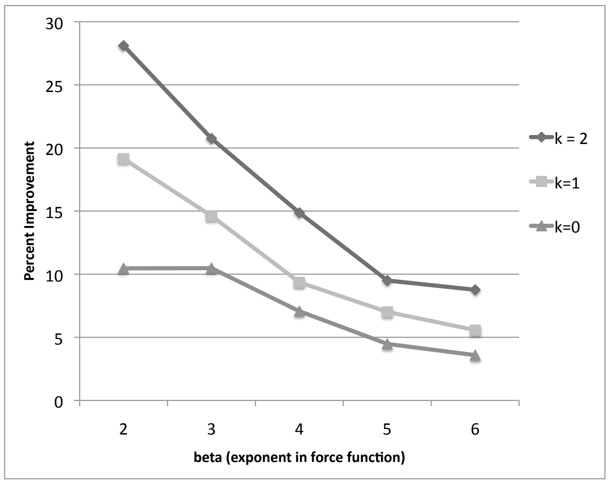

Figure 3: Percent Improvement when $n = 40$, $m = 20$

result, improving on the Naïve algorithm output by some percentage points is still significant. For the simulations then we could expect to see values of percent improvement close to this one if the GPA is performing well.

6.2 Results

Figures 3 to 5 show the results of the simulations when $n = 40$ for different values of $m \in \{20, 30, 40\}$. Each figure shows the results for the percent improvement by using the GPA over the NPA. Each line represents the results for a given value of regional skew for the locations of the slots.

We can see that when choosing a β value between 2 and 6 the GPA improved upon the average distance for the NPA no matter what the value of regional skew was. In all cases the greater improvement was seen when the regional skew was 2. The worse improvement was seen when the regional skew was 0 (uniform distribution) in all cases as well. In most cases, the value of β (the exponent of the force function) that yielded the best results was $\beta = 2$ and for some cases $\beta = 3$.

Similar trends are seen in the results for $n = 80$ (Figure 6). For these tests the number of vehicles were increased to $n = 80$ and the tested number of available slots were $m \in \{40, 60, 80\}$. The results for $n = 80$ and $m = 40$ or 80 were similar to the figures obtained for $n = 40$ and were left out for space constraints. Choosing a β value between 2 and 6 showed improvement in all cases again. The improvements were more significant for higher values of skew ($k = 2$).

The results show how all vehicles acting independently are better off using some form of the Gravity-based Parking algorithm (with $\beta = 2$ or $\beta = 3$ if the regional skew is closer to 0) since on average they will see improvement when choosing to move based on gravitational pull of the slots rather than simply moving greedily to the closest slot by using the Naïve Parking algorithm.

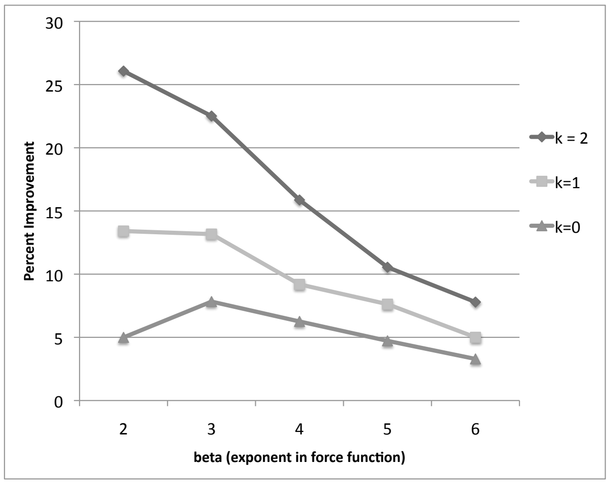

Figure 4: Percent Improvement when $n = 40$, $m = 30$

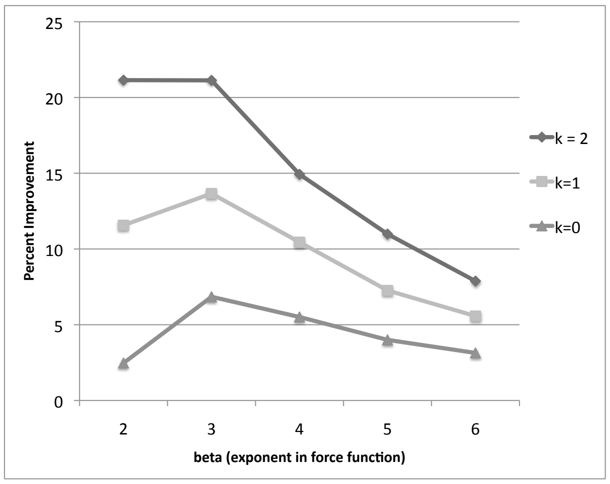

Figure 5: Percent Improvement when $n = 40$, $m = 40$

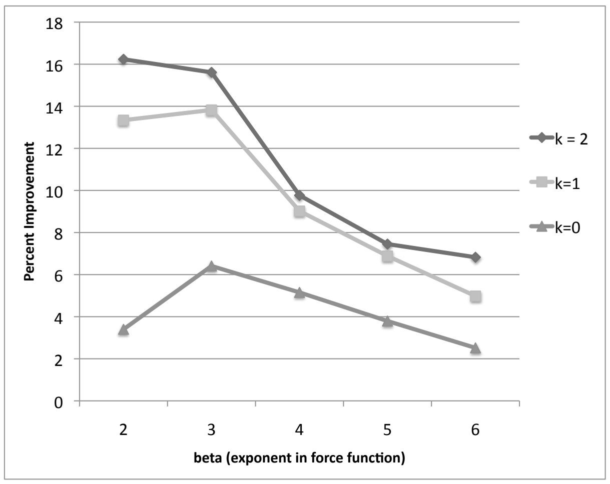

Figure 6: Percent Improvement when $n = 80$, $m = 60$

The results showed improvements in all tested cases. In cases of regional skew of 0, the obtained percent improvement was marginal. But in cases with high regional skew the results were very favorable, in many cases showing an improvement of more than 25%. This is more than five times the expected improvement obtained analytically from the examples presented in section 5.2. A 25% improvement for the time taken to find a parking slot (proportional to the tested distance traveled because of constant velocity), would reduce vehicle miles traveled for a city like Chicago by 15.75 million VMT (according to analysis presented in section 1). This gives a reduction of over 785,000 gallons of gasoline and of over 12,000 tons of CO_2 emissions per year.

7. RELATED WORK

Approaches for monitoring and sensing parking spaces have been presented recently. In this paper, we've assumed that these works exist and that vehicles can receive information about open parking spaces at any time. In [14], ultrasonic sensor technology is used to determine the spatial dimensions of open parking spaces. Wireless sensors are used in [12] to track open parking spaces in a parking facility. These works show how one can detect open spaces. In [9], detection is coupled with sharing of the parking space information in a mobile sensor network. Mathur et. al. present a methodology for vehicles driving past curbside parking spaces to detect open ones, as opposed to having to spend on equipping each parking space with wireless sensors for monitoring. These mobile sensors generate a map view of parking space availability.

Work has been performed on dissemination of reports of open parking spaces [23]. In [23], a parking choice algorithm is presented that chooses parking spaces based on a relevance metric that includes the age of the open parking report. Their work assumes that a driver knows the expected time a slot will remain available from now, and it knows how long it will take to travel there. In our context, it is as if a driver d knows what is the probability that another driver will get there before d. This is a strong assumption that we do not make in our work. Furthermore, the focus of [23] was on peer-to-peer dissemination of parking reports.

The value of having parking information is tested in a peer-to-peer environment in [7]. Kokolaki et. al. show through simulations how vehicles with access to information about open parking spaces have an advantage over vehicles that don't.

Wireless Ad-hoc networking is also used in [22] to search for open parking spaces. They present an algorithm based on the time-varying Traveling Salesman Problem to compute a tour of the open spaces in order for each vehicle to search for parking in the order of the computed tour. Like in [23], their approach depends on knowing the probability that the parking space will still be open after some time. Furthermore, in [19], the relevance of parking reports in a Vehicular Ad-hoc Network is studied.

In [3] and [5], reservation systems for parking spaces are studied. A centralized reservation system is presented in [3]. A server collects information from road-side units and other vehicles and reserves spaces for vehicles. In [5], the reservation system is distributed amongst peers in a Vehicular Ad-hoc Network. They reserve spaces by requesting spaces to a specified peer called the coordinator for each slot. These systems attempt to circumvent the competition for parking spaces by using reservations. In our work we analyze parking competition by using game theory. Indeed existing parking systems are competitive rather than reservation-based.

In [21], the problem of matching spatial datasets is considered. Theirs is a more general version of our problem which would be akin to modeling our parking problem with parking lots. They compute an assignment that minimizes distance traveled but with other added capacity constraints that we don't consider here. Their approach is one applicable for the centralized parking problem only.

The gap between the Nash equilibrium and system optimal assignments and the price of anarchy has been studied for other transportation applications. For example, in [16], the authors show that the POA in the static vehicular routing problem has a bound of 4/3. In the static routing problem, vehicles with known origins and destinations are assigned paths on a network in order to minimize some system wide objectives. Dynamic vehicular routing is similar to its static counterpart but it includes a notion of time that the static problem lacks. In the dynamic problem, link travel times are affected by the number of vehicles on the links at each time point. Whereas in the static problem the travel time on each link is affected by all vehicles that pass through it at any point during the time horizon being considered, regardless of at what specific time the vehicles passed through the link. In [2], the authors prove that the POA is unbounded for dynamic vehicular routing problems. In our work we also showed that the POA is unbounded albeit for a different problem, namely the vehicular parking problem.

Our tested algorithm is based on using gravitational force to model the attractiveness of parking regions. Gravity models have been employed in other computing applications that use Euclidean data. For example, in [6, 24], gravity models are used for Euclidean data clustering.

8. CONCLUSION

In this paper our main goal was to analyze vehicular parking. We presented various models or contexts in which the parking problem was studied.

In the centralized model we show how a centralized authority would assign vehicles to parking slots in order to minimize the system-wide and total distance traveled by all vehicles. We also presented an algorithm for computing this assignment of polynomial complexity.

For the more realistic distributed case we introduced the Parking Slot Assignment Game (Psag). For the complete information model, we defined the Nash Equilibrium for the game and showed that all instances of complete information Psag have a Nash equilibrium in pure strategies (strategies that do not randomize). We also presented an algorithm for computing this Nash equilibrium strategy for each player. For this model we also presented a proof that the price of anarchy for this problem is in general unbounded.

In the incomplete information case, no player has the luxury of knowing the locations of other players. For this problem we presented two simple examples for which we were able to compute the Nash equilibrium. These examples served as a jump-off point to introduce the Gravity-based Parking Algorithm. Simulations were run to test the performance of this algorithm against the Naïve Parking algorithm and the results showed an improvement for players when choosing to use the Gravity-based algorithm over the Naïve algorithm (greedy) in all of the tested cases. The GPA showed improvements of more than 25% for various tested cases. This is an improvement that would reduce 785,000 gallons of gasoline waste and 12,000 tons of CO_2 emissions per year on a big city like Chicago.

9. ACKNOWLEDGEMENTS

This work was supported by grant INSERT NUMBER HERE. We would also like to thank Prof. T.E.S. Raghavan for helpful discussions on the game-theoretic aspects of our work.

10. REFERENCES

[1] http://sfpark.org/.

[2] E. Anshelevich and S. Ukkusuri. Equilibria in dynamic selfish routing. In *Proc. of 2nd Intl. Symp. on Algorithmic Game Theory (SAGT)*, 2009.

[3] J. Boehlé, L. Rothkrantz, and M. van Wezel. Cbprs: A city based parking and routing system. *ERIM Report Series Reference No. ERS-2008-029-LIS*, May 2008.

[4] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* John Wiley & Sons, 1998.

[5] T. Delot, N. Cenerario, S. Ilarri, and S. Lecomte. A cooperative reservation protocol for parking spaces in vehicular ad hoc networks. In *Proc. of the 6th Int. Conference on Mobile Technology, Application and Systems*, Nice, France, September 2009.

[6] J. Gomez, D. Dasgupta, and O. Nasraoui. A new gravitational clustering algorithm. In *Proc. of SIAM Conf. on Data Mining (SDM)*, pages 83–94, 2003.

[7] E. Kokolaki, M. Karaliopoulos, and I. Stavrakakis. Value of information exposed: wireless networking solutions to the parking search problem. In *Proc. of 8th Intl. Conf. Wireless On-Demand Network Systems and Services (WONS)*, Bardonecchia, Italy, January 2011.

[8] J. R. Marden and T. Roughgarden. Generalized efficiency bounds in distributed resource allocation. In *49th IEEE Conf. on Decision and Control (CDC)*, pages 2223–2238, Atlanta, GA, Dec. 2010.

[9] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrashekharan, W. Xue, M. Gruteser, and W. Trappe. Parknet: Drive-by sensing of road-side parking statistics. In *MobiSys*, San Francisco, CA, June 15-18 2010.

[10] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

[11] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani. *Algorithmic Game Theory.* Cambridge University Press, New York, NY, USA, 2007.

[12] B. Panja, B. Schneider, and P. Meharia. Wirelessly sensing open parking spaces: Accounting and management of parking facility. In *AMCIS 2011 Proceedings*, 2011.

[13] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., 1982.

[14] W. Park, B. Kim, D. Seo, D. Kim, and K. Lee. Parking space detection using ultrasonic sensor in parking assistance system. In *IEEE Intelligent Vehicles Symposium*, Eindhoven, The Netherlands, June 2008.

[15] E. Rasmusen. *Games and Information.* Blackwell Publishing, 4th edition, 2006.

[16] T. Roughgarden and E. Tardos. How bad is selfish routing? *J. of the ACM*, 49(2), March 2002.

[17] D. Shoup. *The High Cost of Free Parking.* American Planning Association, Chicago, IL, 2005.

[18] D. Shoup. Cruising for parking. *Transport Policy*, 13:479–486, 2006.

[19] P. Szczurek, B. Xu, J. Lin, and O. Wolfson. Spatio-temporal information ranking in vanet applications. *Intl. J. of Next-Generation Computing*, 1(1), July 2010.

[20] Transportation Alternatives (www.transalt.org), New York, NY. *Pricing the Curb: How San Francisco, Chicago and Washington D.C. are reducing traffic with innovative curbside parking policy*, July 2008.

[21] L. H. U, K. Mouratidis, M. L. Yiu, and N. Mamoulis. Optimal matching between spatial datasets under capacity constraints. *ACM Trans. on Database Systems*, 35(2), 2010.

[22] V. Verroios, V. Efstathiou, and A. Delis. Reaching available public parking spaces in urban environments using ad-hoc networking. In *IEEE Intl. Conf. on Mobile Data Management (MDM)*, 2011.

[23] O. Wolfson, B. Xu, and H. Yin. Dissemination of spatio-temporal information in mobile networks with hotspots. In *Proc. of the 2nd Intl. Workshop on Databases, Information Systems, and Peer-to-Peer Computing*, pages 185–199, Toronto, Canada, 2004.

[24] W. Wright. Gravitational clustering. *Pattern Recognition*, 9:151–166, 1977.