Multi-Agent Parking Game

Giuseppe Calise , Aniello Murano , Silvia Stranieri
University of Naples Federico II, Italy
calisegiuseppe@outlook.com, nello.murano@gmail.com, silvia.stranieri@unina.it

Abstract

In this paper, we propose a game-theoretic solution to the parking problem, by exploiting a strategic-reasoning approach for multi-agent systems. Precisely, cars are modeled by agents interacting among them in a multi-player game setting, whose aim is to get a free slot parking-place satisfying their own constraints. The overall assignment is then given as a Nash equilibrium solution. We come up with an algorithm (and its implementation in a tool) that works in quadratic time. We give evidence of the benefits of our approach by running our tool on a large hospital parking space.

1 Introduction

With the fast development of economy and city modernization, traffic congestion and parking have become serious social problems. Studies conducted in big cities report that daily, on average, drivers take more than eight minutes to park, causing the 30% of traffic [Ayala *et al.*, 2011; Shoup, 2005]. Such statistics raise several side effects, among which a high fuel consumption, high CO_2 emissions, but also a stressful lifestyle for drivers. The growth of Artificial Intelligence applications to automotive is constantly increasing the request for smart solutions to parking. This research field is well identified as *smart parking* (see [Lin *et al.*, 2017]).

The competitive nature of the parking process, during which the drivers compete in order to get an available parking slot for their cars, is the inspiration of this work. Indeed, by exploiting basic settings of the strategic reasoning for multi-agent systems, we model the parking process as a competitive multi-player game in which each car is an agent interacting with all the other ones, with the ultimate goal of getting an available slot that satisfies its own constraints. The *parking problem* we face concerns parking as many cars as possible, while satisfying their requirements.

A multi-agent system is made of autonomous entities, with distributed information, computational capabilities, and possibly diverging interests. These kind of systems have been exploited in several fields: electronic [Song *et al.*, 2021] and industrial process control [Bakliwal *et al.*, 2018], economy [Pal *et al.*, 2018], home automation [Zouai *et al.*, 2017], open system verification [Alur *et al.*, 2002], to name a few.

The way autonomous agents can interact with each other can be classified into two categories: competitive and cooperative. In the former case, there is no a-priori agreement among agents, as they try to maximize their own objective, no matter what the objectives of the other agents are. In the latter case, the agents coordinate among them in order to get the best outcome possible for everyone [Wooldridge, 2009].

Our contribution. We address the parking problem by means of a multi-agent strategic-reasoning approach. Specifically, we model the parking problem as a multi-agent game where cars are competitive agents, moving concurrently and under perfect information. We assume that each agent comes with a desired time-limit to accomplish the car parking. Then, for an agent, the choices are strategies whose payoff reflects the maximum time he consumes to park his own car (or the fact that he cannot park at all). Solving the parking problem corresponds to finding a solution in such a multi-agent game that minimizes all agents' payoffs, in a fair way. We impose such a fairness by means of Nash equilibrium and prove its effectiveness on real scenarios. We recall that in a game a Nash equilibrium is reached when each player does not have any incentive to unilaterally change his strategy.

The contribution of this work is twofold. From one side, we come up with an effective multi-agent game model for the parking problem we consider. From the other side, we provide an algorithm (and its implementation in a tool), working in quadratic time, that allows a fair allocations of the parching slots by satisfying a Nash equilibrium. As we prove later on practical scenarios, this is a valuable compromise with respect to an optimal, but exponential, brute-force solution that would check all possible distribution of cars over available slots. Also, as expected, our solution is better than any naive FIFO approach. Indeed, consider a scenario in which there are three vehicles, V_1, V_2, and V_3, looking for a parking, and three slots available A, B, and C. Assume now that V_1, V_2, and V_3 have up to 7, 5, and 3 minutes to accomplish the parking, respectively. Also, assume that slots A, B, and C require 2, 3, and 5 minutes to be reached, respectively. Assume now that V_1 picks A and V_2 picks B; then, V_3 would not have enough time to reach the remaining slot C. Contrarily, a solution that allows parking all vehicles by accommodating their requirements is to assign V_1, V_2, and V_3 to C, B, and A, respectively. This is exactly what our algorithm would return as a solution.

2 Related Works

Smart parking solutions literature is very reach and diversified. In [Lin *et al.*, 2017], the authors provide a large survey on smart parking modeling, solutions, and technologies as well as identify challenges and open issues. Algorithmic solutions have been also proposed in the VANET research field, see for example [Senapati and Khilar, 2020; Rad *et al.*, 2017; Safi *et al.*, 2018; Balzano and Stranieri, 2019; Balzano *et al.*, 2016; Balzano *et al.*, 2017].

Less common is the use of game-theoretic approaches to address the parking problem. An exception is [Kokolaki *et al.*, 2013], which is probably the closest to us. Here the authors also propose a parking solution based on the Nash equilibrium. However, differently from us, they provide a numerical solution (rather than an algorithm or a tool), and, more importantly, they consider a scenario with both private and public parking slots, and the drivers' payoffs strongly rely on such a topology; it is not clear to us how to massage their model to accommodate our setting.

Many other smart parking mechanisms have been proposed in the literature based on a multi-agent game setting. In [Małecki, 2018], drivers' behavior is simulated by modeling the environment on the basis of cellular automata. In [Belkhala *et al.*, 2019] the model is based on the interaction between the user (driver) and the administrator, but focusing more on the architecture rather than the model setting and the strategic reasoning. Similarly, [Jioudi *et al.*,] provides an E-parking system, based on multi-agent systems aimed to optimize several users' preferences. In [Okoso *et al.*, 2019], the authors manage the parking problem with a cooperative multi-agent system, by relying on a priority mechanism. In [Pereda *et al.*, 2020], the authors also focus on an equilibrium notion, but they study the Rosenthal equilibrium rather than the Nash one, which describes a probabilistic choice model. Finally, in [Lu *et al.*, 2021] the authors also consider a Nash equilibrium applied to cars, but they only use it to talk about traffic, rather than parking.

In this work, not only we design the parking process as a game among agents playing competitively, but also we study the use of the Nash equilibrium as a fair solution. To the best of our knowledge, this is the first work addressing the parking problem via multi-player game, whose solution is given algorithmically by solving a Nash equilibrium.

3 A Real Scenario

As case study, we have focused on the parking area of the *Federico II Hospital Company* in Naples, one of the biggest and most specialized hospital in the South of Italy, whose construction goes back to the early Sixties.

The Hospital, as it is schematically depicted in Figure 1, is made of 21 building blocks, distributed over $440.000m^2$, and provides in total one thousand of beds for ordinary recovery and two hundreds of beds for day-hospital use. The parking space, having 2684 slots in total, consists of 21 independent areas, and is mainly used by patients and, in turn, by the 3400 employees (doctors, nurses, technicians, administrators, etc.).

The hospital has four guarded gates, one of which is for pedestrian. The car gates are preceded by a road where cars line up for the necessary checks. In average, it is estimated that there are 4.600 car accesses per day. There is no policy about the allocation of the parking places and, except for few reserved ones, each driver chooses by its own the slot. This disorganized solution produces a huge traffic congestion, bottlenecks at the entrance, and an unbalanced distribution of the cars over the parking area. More importantly, it does not take into account the specific constraints and some physical limitations of the users, such as walking issues or urgency. In the most crowded hours, in average, the drivers spend more than 20 minutes to find a parking slot or, even worst, they leave the parking area by missing available slots.

In order to efficiently apply our tool, we assume that the list of available slots in every area of the hospital is known at runtime. Also, we make use of all information the car passengers have to pass to the hospital before entering, and in particular their logistics. Finally, we assume that the drivers will be followed while driving inside the parking area, by means of tracking devices (GPS, smartphone, videocameras, etc.).

Figure 1: Graphic representation of the A.O.U. Federico II

Having all this information at his disposal, the tool works as follows: it takes all cars in queue on the roads in front the car gates, as well as all the specific needs and constraints of their occupants. Then, it processes the data and following the algorithm described in the sequel, it opportunely associates the available slots to the cars. In particular, the tool will access both the Employers Data Center and the Online Booking Center of the hospital and, thanks to the latter, the tool will know which kind of services the patients need, date and time of their appointments, possible walking limitations and handicaps, etc. Finally, note that the tool operates in stages, processing one bunch of cars at the time, as they are in queue. Someone may criticize this solution and propose an offline allocation instead. We decide not to follow this solution for two main reasons: first, the hospital is highly dynamic in slot requests and, more importantly, slots are very limited in numbers, so it is better to allocate slots only when cars show up.

4 Parking Game Structure

In this section we introduce the *Parking Game Structure* model, (*PGS*, for short), that is the game model we will use along our algorithm to reasoning about the parking problem we address. The model definition takes inspiration from the scenario described in Section 3. Thus, in a PGS, the players are cars with their needs and constraints. Also, the PGS takes into account all the specification about the slots, in particular

their location, their availability, the time they require to be reached from the entrance, and so on.

Formally the *Parking Game Structure* is defined as follows:

Definition 4.1 (Parking Game Structure)**.** *The Parking Game Structure (PGS) is a tuple:*

$$\mathcal{G} = (Agt, S, G, g, F, T, R) \quad (1)$$

where:

- $Agt = \{a_1,...,a_n\}$ *is a set of* agents*, i.e., the cars,*
- $S = \{s_1,...,s_m\}$ *is a set of parking* slots*,*
- $F = \{f_1,...,f_n \mid f_i \in [0,1]\}$ *is a set of* resilience *values,*
- $G = \{g_1,...,g_l\}$ *is a set of* gates*,*
- $g : Agt \to G$ *is a function associating agents to gates,*
- $AT = \{t_1,...,t_n\}$ *is a set of* agent-time *values, where* t_i *is the time limit the car* a_i *has for parking,*
- $RT = \{r_{(1,1)},...,r_{(m,l)}\}$ *is a set of* reaching-time *values, where* $r_{(i,j)}$ *is the time needed to reach the parking slot* s_i *from gate* g_j*.*

Regarding the set of resilience indexes F, note that each f_i is associated with agent a_i and it has a twofold use: first, it imposes an order among agents; second, it affects the final pre-emption order. This will be more clear below. For simplicity, we assume that all the resilience indexes are different, i.e., $f_i \neq f_j$ for every $1 \leq i < j \leq n$. The indexes in F can be set manually as input, however we report that, for the case study we have introduced in Section 3, the values have been obtained automatically by processing the information coming from the Employers Data Center and the Online Booking Center of the hospital; in particular, for the patients, the resilience index represents their movement ability, therefore, the lower the rate, the more favored the patient.

A *strategy* for an agent a_i consists of choosing a slot $s_j \in S$. Formally it is a function $Str : Agt \to S$. A *strategy profile* is an n-uple $\overline{s} = (\overline{s_1},...,\overline{s_n})$ of strategies, one for each player. Formally, in $\overline{s}$, for each i, we have $Str(a_i) = \overline{s_i}$. It is worth noting that it may happen that two or more players choose the same strategy. Next we define the *costs associated* to $\overline{s}$ as a tuple of costs $\overline{c} = (\overline{c_1},...,\overline{c_n})$. Then, a *payoff* π of a strategy $\overline{s}$ is defined as a sum of all such $\overline{c_i}$, i.e., $\pi(\overline{s}) = \sum_i \overline{c_i}$, and by π_i we denote the $i-$th cost value of that tuple.

Definition 4.2. *Let* $a_i \in Agt$ *be an agent with* $g(a_i) = h$ *and* $\overline{s} = (\overline{s_1},...,\overline{s_n})$ *be a strategy profile, with* $\overline{s_i} = s_j$ *for an* $s_j \in S$*. We define the* costs associated *to* $\overline{s}$ *as the tuple* $\overline{c} = (\overline{c_1},...,\overline{c_n})$ *where each* $\overline{c_i} =$

$$\begin{cases} f_i \cdot (t_i - r_{(j,h)}) & \text{if 1. } (t_i - r_{(j,h)}) \geq 0\text{, and} \\ & \text{2. there is no } a_{k \neq i} \text{ such that} \\ & f_k < f_i\text{, } \overline{s_k} = \overline{s_i}\text{, and} \\ & (t_k - r_{(j,p)}) \geq 0\text{, with } g(a_k) = p \\ \infty & \text{otherwise} \end{cases} \quad (2)$$

In words, the value $\overline{c_i}$ is a finite value if the agent a_i has enough time to reach the parking slot s_j and such a slot has not been taken from any other agent a_k with a lower resilience (i. e., $f_k < f_i$). Then, the value, when it is finite, reflects how much time it is left to the agent after he has reached the assigned slot (with respect the total time he has at his disposal). Conversely, the infinity value corresponds to the worst possible outcome for the agent a_i, which reflects the fact that he cannot park at the slot s_j. At this point, it should be intuitive that the problem of looking for an optimal strategy profile $\overline{s}$ can be reduced to the problem of minimize[1] the corresponding vector of associated costs $\overline{c}$. Unfortunately, this is in general not an easy task. In particular, a brute-force algorithm checking all the possible strategy profiles is unfeasible as it requires exponential-time. Conversely, we suggest adopting a Nash equilibrium solution which provides, by definition, a fair solution and, along with our setting, it just requires quadratic-time. In the sequel we are going to present such a solution. Also, we present a solution based on a naive (greedy) behavior of the players, which reflects the current behaviour of drivers at the parking of the hospital described in Section 3: each car takes the first available parking slot which satisfies its needs. By means of a toy example, we show that the solution based on the Nash equilibrium over-perform the one based on the greedy behaviour of the players.

5 The Parking Slot Selection Game

In this section, we first provide a toy example, then we introduce the *Parking Slot Selection Game* (PSSG, for short) and propose a solution by means of a Nash equilibrium calculation. We also comment on the greedy (naive) approach and compare it with our solution. For a matter of presentation, we will recall the notion of Nash equilibrium.

W.l.o.g., in the sequel we restrict to parking structures having just one gate. This means that we can get rid of g and G in PGS as given in Definition 4.1, as well as the second index of the reaching-time values in RT. This also simplifies the definition of costs associated with strategy profiles.

5.1 A Running Example

Let us consider a parking place with 3 slots available and 3 cars aiming at parking. Let us suppose that the first, the second, and the third car have respectively 5, 2, and 4 minutes available to park and that, as associated resilience they have 0.5, 0.1, and 0.009, respectively. Also, suppose that the first,

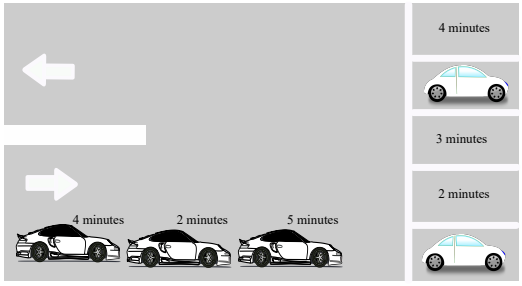

Figure 2: 3-players-3-slots Game.

[1]Note that the minimization guarantees that the best slots are kept for future use, so to focus on the continue allocation process rather than the single stage. At any rate, one can use maximization (in a finite domain) without affecting the rest of the algorithm.

the second, and the third slot require 2, 3, and 4 minutes to be reached, respectively. We call such a game the 3-players-3-slots game and it is reported in Figure 2.

5.2 The Price of Anarchy: a Naive Solution

When a car is approaching to the parking, a naive solution is to occupy the first slot it can get. This approach leaves to the car a free will to park in the slot that best fits its constraints, without paying attention to the other cars requirements. This easy-to-design solution may lead to a non optimal vehicles allocation, as it may leave out some cars (not able to park), as the remaining slots may not satisfy their requirements.

To give an example, let us consider the scenario described in Section 5.1. In this situation, the first car would choose the closest slot (the one that requires 2 minutes to be reached). Then, the second car would not be able to park, because all the remaining free slots are too expensive in terms of time. For this reason, we have looked for a better solution that would exploit the car parking potentialities at the best by means of a fair distribution of slots among cars, and that would be computationally easy to be calculated on the fly.

5.3 Nash Equilibrium Based Solution

In game theory, a well-conceived solution concept that ensures a robust form of fairness and satisfaction among players is *Nash equilibrium*. This concept was deeply investigated and well formalized by John Nash in the fifties, both under pure and mixed strategies (see [Van Damme, 1991] for more details). In the basic definition, we say that in a multi-player game, all players, moving concurrently, reach a Nash equilibrium if none of them has the incentive to unilaterally deviate from that equilibrium. By casting this in our parking scenario, we try to reach a situation in which all drivers are associated to parking slots, by means of an equilibrium over their constraints. In other words, our goal is to provide a strategic profile (parking slot assignment) in which no player wants to change his slot unless some other players want to change theirs.

Following the model definition given in Definition 4.1 and the observations made above, we formally introduce the *Parking Slot Selection Game* we address, as follows.

Definition 5.1 (Parking Slot Selection Game)**.** *The Parking Slot Selection Game (PSSG) has an input and an output defined as follows:*

- *Input: a PGS $\mathcal{G}$, as given in Definition 4.1.*
- *Output: a strategic profile $(\overline{s}_1^*,...,\overline{s}_n^*)$ providing a Nash equilibrium for $\mathcal{G}$. A strategy profile $(\overline{s}_1^*,...,\overline{s}_n^*)$ is a Nash equilibrium iff $\forall\, \overline{s}_1,...,\overline{s}_n \in S$ it holds that:*

$$\pi_1(\overline{s}_1^*,...,\overline{s}_n^*) \leq \pi_1(\overline{s_1}, \overline{s}_2^*..,\overline{s}_n^*)$$
$$\pi_2(\overline{s}_1^*,...,\overline{s}_n^*) \leq \pi_1(\overline{s}_1^*, \overline{s_2}..,\overline{s}_n^*)$$
$$\vdots$$
$$\pi_n(\overline{s}_1^*,...,\overline{s}_n^*) \leq \pi_1(\overline{s}_1^*, \overline{s}_2^*..,\overline{s_n})$$

In words, the PSSG consists in looking for a strategy profile that, with respect to the associated costs, no players has an incentive to unilaterally change his choice.

Similarly to the PSGG, one can define the *Naive Parking Game* (*NPG*, for short). To give some details, first assume that in an NPG players are ordered, then the strategy profile $(\overline{s}_1^*,...,\overline{s}_n^*)$ is such that for each agent a_i, it holds that $\overline{s}_i^*$ is the best choice (in terms of minutes to reach it) over $S \setminus \{\overline{s}_1^*,...,\overline{s}_{i-1}^*\}$.

5.4 A Solution to the 3-players-3-slots Game

Let us consider again the 3-players-3-slots game described in Section 5.1. We now show a solution based on the satisfaction of a Nash equilibrium. As we will see in a while, such a solution allows accommodating all cars, while satisfying all their constraints, contrarily to what we have seen with the naive solution. Later, we will show that this is true in general and not just for the case of our specific example. Let us formally describe the 3-players-3-slots example by means of a PGS $\mathcal{G}_3$ whose components are defined as follows:

- $Agt = \{car_1, car_2, car_3\}$ is the set of cars,
- $S = \{slot_1, slot_2, slot_3\}$ is the set of parking slots,
- $AT = \{5, 2, 4\}$ is the set of time-values car_1, car_2, and car_3 have at at their disposal, respectively,
- $RT = \{2, 3, 4\}$ is the set of times needed to reach the slots $slot_1$, $slot_2$, and $slot_3$, respectively,
- $F = \{0.5, 0.1, 0.009\}$ is the set of cars resilient values,
- The cost function is reported in Table 1, in the last three rows. For instance, the triple $(\infty, \infty, 0.018)$ represents the case in which all cars decide to park in the same slot $slot_1$; so, car_3, which has the lowest resilience value, gets it at a cost of 0.018 (i.e., $(4-2)\cdot 0.009$), while the other cars leave the process incomplete, as they get ∞.

By a matter of calculation, one can check that there exists only one Nash equilibrium, which corresponds to $\overline{s} = (slot_2, slot_1, slot_3)$, with $\overline{c} = (1, 0, 0)$ (in bold in Table 1), and $\pi(\overline{s}) = 1$.

5.5 A Solution to the Parking Slot Selection Game

In this section, we introduce the algorithm for the solution to the problem described in Definition 4.1. We first provide the pseudo-code in Algorithm 1, then we describe how it works and report on its time complexity.

With the first iteration, the car with the lowest resilience index, *actualCar*, is selected from the queue, through the function *priorityCar(·)*, which takes as input the set of cars and returns the one with the lowest resilience index respect to the others. The variable cost *outcome* is associated an infinity value, the worst possible one. In the second iteration, the algorithm computes the costs resulting from the function *c(·)*, which takes as input a car and a slot. The value of the *outcome* is updated with the value of the best cost computed. Among the available slots, the one with the best result is assigned to the *actualCar*. Once assigned, the slot is remove from the set of the available ones, with the function *setNotAvailable(·)*.

Theorem 5.1 (Correctness of Algorithm 1)**.** *Algorithm 1 computes the Nash equilibrium for the game.*

		car_3								
		$slot_1$			$slot_2$			$slot_3$		
		car_2			car_2			car_2		
		$slot_1$	$slot_2$	$slot_3$	$slot_1$	$slot_2$	$slot_3$	$slot_1$	$slot_2$	$slot_3$
car_1	$slot_1$	$\infty, \infty, 0.018$	$\infty, \infty, 0.018$	$\infty, \infty, 0.018$	$\infty, 0, 0.009$	$1.5, \infty, 0.009$	$1.5, \infty, 0.009$	$\infty, 0, 0$	$1.5, \infty, 0$	$1.5, \infty, 0$
	$slot_2$	$1, \infty, 0.018$	$1, \infty, 0.018$	$1, \infty, 0.018$	$\infty, 0, 0.009$	$\infty, \infty, 0.009$	$\infty, \infty, 0.009$	**1,0,0**	$1, \infty, 0$	$1, \infty, 0$
	$slot_3$	$0.5, \infty, 0.018$	$0.5, \infty, 0.018$	$0.5, \infty, 0.018$	$0.5, 0, 0.009$	$0.5, \infty, 0.009$	$0.5, \infty, 0.009$	$\infty, 0, 0$	$\infty, \infty, 0$	$\infty, \infty, 0$

Table 1: Cost function values for 3-drivers-3-slots instance of the game.

Algorithm 1 Algorithm for the solution of the PSSG.

Input: Queue of ready vehicles
Output: Slot allocation

1: **while** $carQueue \neq null$ **do**
2: $actualCar = priorityCar(carQueue)$.
3: $outcome = \infty$.
4: **for** $slot \in setAvailableSlots$ **do**
5: $po = c(actualCar, slot)$.
6: **if** $po \geq 0$ & $po < outcome$ **then**
7: $outcome = po$.
8: $assignSlot(actualCar, slot)$
9: $setNotAvailable(slot)$.
10: **end if**
11: **end for**
12: **end while**

Proof (Sketch). Proving that the algorithm provides a Nash equilibrium is quite trivial. Assume by contradiction that $\overline{s} = (\overline{s_1}, .., \overline{s_n})$ is the solution provided from our algorithm and it is not a Nash equilibrium. Then, by definition of Nash equilibrium, there must exist an agent, let us say agent a_i, whose strategy s_j is not the best, while fixed the strategies for the other players. Hence, there exists another strategy s'_j for the agent a_i, such that the payoff of s'_j is better than the one for s_j (given the same strategies for the other players). But if such a strategy s'_j exists, then it would be found at row 6 of our algorithm, and it would be chosen as the final strategy for agent a_i. But this clearly contradicts the hypothesis that $\overline{s} = (\overline{s_1}, .., \overline{s_n})$ is the solution provided. □

Theorem 5.2 (Complexity of Algorithm 1)**.** *The complexity of Algorithm 1 is quadratic with respect to the number of agents involved in the game, in the worst case.*

Proof. Consider the worst possible scenario, i.e., no vehicle obtains a parking slot. Then, let us compute $\mathcal{C}(PSSG)$ as the complexity of the Parking Slot Selection Game. The proof proceeds by analyzing the complexity of the most expensive operations, from the inner ones to the outer ones. We use the notation $\mathcal{C}(r)$ to indicate the complexity of the code from the r-th row of the Algorithm 1.

The function $assignSlot(Car, slot)$ performs simple assignments, with a constant complexity $\mathcal{C}(7) = O(1)$.

The inner loop does not perform any slot assignment, in the considered worst case, since none of them satisfies the constraints of the cars to be allocated. Hence, the inner loop is repeated as many times as $|S|$, where S is the set of slots, according to Definition 4.1. Assuming that $|S| = m$, we can deduce that $\mathcal{C}(3) = \sum_{i=1}^{m} \mathcal{C}(7) = \sum_{i=1}^{m} O(1) = O(m)$.

The outer loop is performed as many times as the number of cars, i.e, the agents. Since by Definition 4.1 $|Agt| = n$, we have that $\mathcal{C}(1) = \sum_{j=1}^{n} \mathcal{C}(3) = \sum_{j=1}^{n} O(k) = O(nm)$.

Assuming that, in the worst case, n and m are of the same order, we can conclude that the total complexity is $\mathcal{C}(PSSG) = O(n^2)$. □

6 Evaluation

In this section, we compare the performances between executing the naive solution to solve NPGs and Algorithm 1 to solve PSGGs. We have run 10 times the two approaches on a growing number of cars and slots. All values and time-limit needed have been generated randomly. Results have been collected in Table 2. Each column represents a different execution of the two approaches with the corresponding input parameters, while the rows keep track of the two analyzed solutions. Each entry contains the number of cars that have been able to park successfully, over the total number of cars involved. As one can observe, the Nash equilibrium based solution is never worse than the naive one. Moreover, by extending the experiment over 100 and 200 executions, our approach is strictly better than the naive one in the 89% and 93% of the cases respectively, and it allocates the same number of vehicles in the remaining ones.

Since, by construction, a greater number of executions determines a greater number of cars, these experiments also prove the scalability of our algorithm, which seems to behave well with high numbers. Such a scalability property will be explained in more details in the next section.

7 Benchmarks

Here we show some benchmarks regarding Algorithm 1. Precisely, we have analyzed the behavior of the algorithm in the management of a growing number of cars waiting for a parking slot, with respect to a fixed number of parking slots. We have considered the following two scenarios:

- In Table 3, the number of slots is 4.600. Such a number is not picked at random, but it refers to the number of slots available inside the structure of our case study.
- In Table 4, the number of slots is 20.000. Also in this case, the number is not picked at random, but it refers to the number of available slots in the biggest parking space of the world (West Edmonton Mall in Canada).

Figures 3 and 4 reflect the quadratic nature of the algorithm: the time is the result of the average between 100 tests. All tests have been executed on an Intel®Core™i5-7300HQ CPU processor of 2,50 GHz, with 8 Gb RAM capacity.

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
	3 slots 3 cars	4 slots 4 cars	5 slots 5 cars	6 slots 6 cars	7 slots 7 cars	8 slots 8 cars	9 slots 9 cars	10 slots 10 cars	11 slots 11 cars	12 slots 12 cars
NPG	3/3	3/4	5/5	6/6	7/7	8/8	7/9	8/10	9/11	12/12
PSSG	2/3	3/4	5/5	5/6	6/7	6/8	6/9	7/10	8/11	10/12

Table 2: Resulting vehicle allocations over 10 different simulations applying two solutions to the parking game: the Nash equilibrium based one, and the naive one.

$slots = 4.000$	
cars	seconds
200	0,001
400	0,002
800	0,004
1.600	0,009
3.200	0,027
6.400	0,402
12.800	1,389
25.600	3,415
51.200	10,165
102.400	33,260
204.800	119,718

Table 3: Results on 4.000 slots.

$slots = 20.000$	
cars	seconds
200	0,003
400	0,006
800	0,013
1.600	0,026
3.200	0,060
6.400	0,150
12.800	0,430
25.600	5,687
51.200	23,597
102.400	57,769
204.800	166,093

Table 4: Results on 20.000 slots.

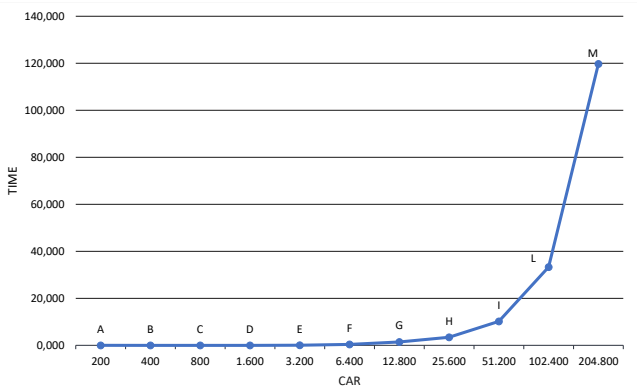

Figure 3: Time and cars variation with 4.000 slots.

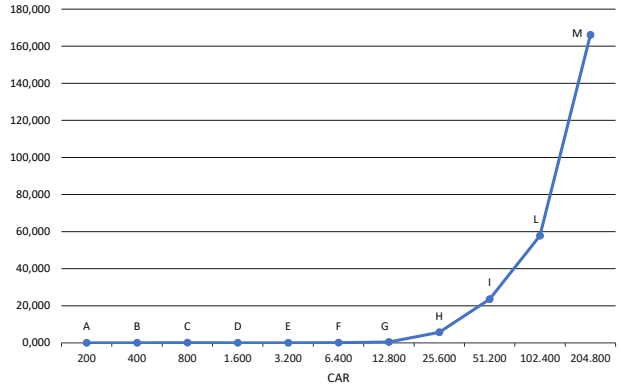

Figure 4: Time and cars variation with 20.000 slots.

To show the good performance of our algorithm, in our benchmarks we have considered a very large set of cars. The benchmarks show that our tool can be also used in other fields, with much higher numbers. For example, it can be used to accommodate people in a stadium, or, distribute people over hospitals, for example, for a massive vaccinations, as it is required nowadays for the Covid pandemic situation.

8 Conclusions and Future Works

The parking problem is one of the most challenging questions in the automotive research field. Inspired by the intrinsic interaction among cars that compete among them in order to get a parking slot complaining with their constraints, in this paper we have explored a game-based approach. Precisely, following a real case study, we have formally introduced (*i*) a multi-player game structure model, (*ii*) the problem, and (*iii*) a solution algorithm working in quadratic time. The game model makes use of costs, reflecting the time ability of a car to park in a specific slot (modulo a resilience grade intrinsically associated to each car). The core of the algorithm is then based on a Nash equilibrium solution, which allows focusing not just on the best choice for a single car, but rather on one that guarantees a fair slot assignment among all cars. The proposed solution requires quadratic time.

We have positively tested our tool on a model of the parking space of the Federico II Hospital in Naples, one of the biggest hospitals in the South of Italy. The construction of the hospital and the annexed parking space goes back to early Sixties. Since there, no parking policy have been ever adopted: excepts for few reserved slots, a car entering the area can park in any slot. This reflects in a serious traffic congestion and an inefficient use of the slots everyday. Conversely, our approach provides, for the first time, a valid and promising solution. In order to put it in practice, we are currently working on a mobile client application to support the drivers along the parking task, from the assignment of the slot while approaching the gate, up to the moment they leave the car.

Our solution is the fruit of a deep analysis of the most parking-congestion-affected sites in our city, together with our strategic reasoning background. Despite being amendable, the provided solution sets the stage for future essential improvements not only of health care services offered by the hospital under exam, but also of facilities from different contexts with similar problems. Simulation results show that our solution improves notably the slot assigning with respect to the naive parking behavior in which each car is free to select a slot according only to its own preferences. The simulation also shows that our tool is scalable and can handle very huge numbers of slots and cars.

References

[Alur *et al.*, 2002] Rajeev Alur, Thomas A Henzinger, and Orna Kupferman. Alternating-time temporal logic. *JACM*, 49(5):672–713, 2002.

[Ayala *et al.*, 2011] Daniel Ayala, Ouri Wolfson, Bo Xu, Bhaskar Dasgupta, and Jie Lin. Parking slot assignment games. In *ACM-GIS*, pages 299–308, 2011.

[Bakliwal *et al.*, 2018] Kshitij Bakliwal, Maharshi Harshadbhai Dhada, Adrià Salvador Palau, Ajith Kumar Parlikad, and Bhupesh Kumar Lad. A multi agent system architecture to implement collaborative learning for social industrial assets. *IFAC*, 51(11):1237–1242, 2018.

[Balzano and Stranieri, 2019] Walter Balzano and Silvia Stranieri. Acop: an algorithm based on ant colony optimization for parking slot detection. In *WAINA*, pages 833–840, 2019.

[Balzano *et al.*, 2016] Walter Balzano, Aniello Murano, and Fabio Vitale. V2V-EN - vehicle-2-vehicle elastic network. volume 98 of *Procedia Computer Science*, pages 497–502. Elsevier, 2016.

[Balzano *et al.*, 2017] Walter Balzano, Aniello Murano, and Silvia Stranieri. Logic-based clustering approach for management and improvement of vanets. *J. High Speed Networks*, 23(3):225–236, 2017.

[Belkhala *et al.*, 2019] Sofia Belkhala, Siham Benhadou, Khalid Boukhdir, and Hicham Medromi. Smart parking architecture based on multi agent system. *IJACSA*, 10:378–382, 2019.

[Jioudi *et al.*,] Bassma Jioudi, Aroua Amari, Fouad Moutaouakkil, and Hicham Medromi. e-parking: Multi-agent smart parking platform for dynamic pricing and reservation sharing service.

[Kokolaki *et al.*, 2013] Evangelia Kokolaki, Merkourios Karaliopoulos, and Ioannis Stavrakakis. On the efficiency of information-assisted search for parking space: A game-theoretic approach. In *IWSOS*, 2013.

[Lin *et al.*, 2017] Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on ITS*, 18(12):3229–3253, 2017.

[Lu *et al.*, 2021] Xiao-Shan Lu, Ren-Yong Guo, Hai-Jun Huang, Xiaoming Xu, and Jiajia Chen. Equilibrium analysis of parking for integrated daily commuting. *Research in Transportation Economics*, 2021.

[Małecki, 2018] Krzysztof Małecki. A computer simulation of traffic flow with on-street parking and drivers' behaviour based on cellular automata and a multi-agent system. *JCS*, 28:32–42, 2018.

[Okoso *et al.*, 2019] Ayano Okoso, Keisuke Otaki, and Tomoki Nishi. Multi-agent path finding with priority for cooperative automated valet parking. In *ITSC*, pages 2135–2140, 2019.

[Pal *et al.*, 2018] Gautam Pal, Gangmin Li, and Katie Atkinson. Multi-agent big-data lambda architecture model for e-commerce analytics. *Data*, 3(4):58, 2018.

[Pereda *et al.*, 2020] María Pereda, J. Ozaita, I. Stavrakakis, and A. Sanchez. Competing for congestible goods: experimental evidence on parking choice. *Scientific reports*, 10(1):1–10, 2020.

[Rad *et al.*, 2017] Farhad Rad, Hadi Pazhokhzadeh, and Hamid Parvin. A smart hybrid system for parking space reservation in vanet. *JACET*, 2017.

[Safi *et al.*, 2018] Qamas Gul Khan Safi, Senlin Luo, Limin Pan, Wangtong Liu, Rasheed Hussain, and Safdar H Bouk. Svps: Cloud-based smart vehicle parking system over ubiquitous vanets. *Computer Networks*, 138:18–30, 2018.

[Senapati and Khilar, 2020] Biswa Ranjan Senapati and Pabitra Mohan Khilar. Automatic parking service through vanet: A convenience application. In *ICCAN*, pages 151–159. 2020.

[Shoup, 2005] Donald Shoup. The high cost of free parking. 2005.

[Song *et al.*, 2021] Xiaoyu Song, Ruopeng Yang, Changsheng Yin, and Bo Tang. A cooperative aerial interception model based on multi-agent system for uavs. In *IAEAC*, volume 5, pages 873–882, 2021.

[Van Damme, 1991] Eric Van Damme. *Stability and perfection of Nash equilibria*, volume 339. 1991.

[Wooldridge, 2009] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[Zouai *et al.*, 2017] Meftah Zouai, Okba Kazar, Belgacem Haba, and Hamza Saouli. Smart house simulation based multi-agent system and internet of things. In *ICMIT*, pages 201–203, 2017.