Multi-Agent Parking Problem with Sequential Allocation

Aniello Murano, Silvia Stranieri and Munyque Mittelmann
University of Naples Federico II, Naples, Italy

Keywords: Autonomous systems, Resource Allocation, Multi-Agent Systems, Nash Equilibrium, Smart Parking.

Abstract: In this paper, we study the multi-agent parking problem with time constraints adopting a game-theoretic perspective. Precisely, cars are modeled as agents interacting among themselves in a multi-player game setting, each of which aims to find a free parking slot that satisfies their constraints. We provide an algorithm for assigning parking slots based on a sequential allocation with priorities. We show that the algorithm always finds a Nash equilibrium solution and we prove its complexity is in quadratic time. The usefulness of our approach is demonstrated by considering its application to the parking area of the Federico II Hospital Company in Naples. Finally, we provide experimental results comparing our algorithm with a greedy allocation and evaluating its performance in the application scenario.

1 INTRODUCTION

With the fast development of economy and the improvement of 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* (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 for this work. Indeed, by exploiting basic settings of strategic reasoning for multi-agent systems, we model the parking process as a competitive multi-player game in which each car is an interacting agent with the ultimate goal of getting an available slot that satisfies its own constraints. In the *parking problem*, we aim at parking as many cars as possible, while satisfying their requirements. This situation can be viewed as a non-cooperative Multi-Agent System (MAS) (Wooldridge, 2009), in which each individual tries to maximize their own objective (getting a parking slot within their time restriction), independently from others' objectives and preferences.

We address the parking problem by means of an approach based on Game Theory and Strategic Reasoning. Specifically, we model this problem as a multi-agent game where cars are competitive agents, acting concurrently and under perfect information. In our setting, each agent has a time constraint denoting the maximum time he can use to park. Also, for each slot, we have a time needed to be reached from each entrance. Then, for an agent, the choices for a slot are strategies whose payoff reflects the 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, that is a strategy profile, that is an equilibrium among the agents. We provide an algorithm that finds such a strategy profile and we show that its solution is a Nash equilibrium. We analyze the algorithm's complexity and compare its performance with a greedy approach for parking selection. Finally, we illustrate our approach and show its effectiveness by considering a real application scenario, specifically, the parking area from the Federico II Hospital Company.

Our Contribution. The contribution of this work is twofold. From one side, we come up with a game-theoretic formalization for the multi-agent parking problem. The main advantage is to provide grounds for analyzing solutions based on strategic reasoning, which we illustrate by considering Nash equilibrium. On the other side, we propose a quadratic time al-

Murano, A., Stranieri, S. and Mittelmann, M.
Multi-Agent Parking Problem with Sequential Allocation.
DOI: 10.5220/0011689800003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 484-492
ISBN: 978-989-758-623-1; ISSN: 2184-433X

gorithm (and its implementation[1]) that finds a Nash equilibrium for the allocation of the parking slots. In our experimental results, the allocation of slots induced by our solution was more efficient than the greedy approach, in terms of the number of cars that successfully find a parking slot. 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. This positive experimental and complexity results show our solution provides a valuable compromise with respect to an optimal, but exponential, brute-force approach that would check all possible distributions of cars over available slots.

Note that the multi-agent game model we set up can also admit more than one Nash equilibrium. In game theory, in general, this is problematic as the players do not know which one to choose. In our setting, however, this is not a problem as it is the allocation sequence will induce a unique strategy profile.

Outline. The paper is organized as follows: We start by presenting the related works in Section 2. Then, in Section 3 we introduce the model and the parking problem at the Federico II Hospital Company as a case study. In Section 4, we propose an algorithm for prioritized multi-agent parking selection and analyze it in terms of complexity and a game theoretic solution concept. In Section 5 we present experimental results in order to (i) compare the performance of the proposed solution in relation to a greedy approach and (ii) benchmark the algorithm in the case study. Finally, Section 6 concludes the paper.

2 RELATED WORK

Smart Parking. 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, 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, indeed 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.

Smart parking mechanisms based on a multi-agent game setting have been also proposed in the literature. 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., 2019) 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, (Lu et al., 2021) also considers the concept of Nash equilibrium applied to cars, but it is used to talk about traffic rather than parking.

Allocation Problems. Our work is also related to the literature on multi-agent resource allocation and sequential mechanisms. Allocation problems are a central matter in MAS in which resources need to be distributed amongst several agents, who may also influence the choice of allocation (Chevaleyre et al., 2006). One setting of this problem is the case of indivisible items (such as allocating parking slots) (Brams et al., 2003). The sequential allocation mechanism is a solution widely studied in the literature (Aziz et al., 2015, 2016; Bouveret and Lang, 2011; Kalinowski et al., 2013a,b; Levine and Stange, 2012) and has been considered in several real-life applications (for instance, to organize draft systems (Brams and Straffin Jr, 1979) and to allocate courses to students (Budish and Cantillon, 2007)). According to a predefined sequence of the agents, this mechanism consists of allowing each agent one by one to pick one item among the remaining ones. Aziz et al. (2015)

[1]The source code is available at https://drive.google.com/file/d/1C-TUntxn3fDJwEbgrgT2fuWb49qjui2A/view?usp=sharing.

investigates sequential allocation mechanisms where the policy for the picking sequence has not been fixed or has been fixed but not announced. Supposing additive utilities and independence between the agents, Kalinowski et al. (2013a) proved that the expected utilitarian social welfare is maximized by the alternating policy in which two agents pick items in a fixed order. The relation between social welfare and choice of policy in this type of mechanism has also been considered (Aziz et al., 2016). The authors explore the case in which a (benevolent) central authority chooses a policy to improve social welfare. In the same setting assuming a benevolent central authority, Bouveret and Lang (2011) showed that the choice of an optimality criterion depends on three parameters: how utilities of objects are related to their ranking in an agent's preference relation; how the preferences of different agents are correlated; and how social welfare is defined from the agents' utilities.

The main advantage of the sequential allocation is its simplicity, both in relation to the protocol and the information requested from agents (i.e., agents do not have to submit cardinal utilities) (Flammini and Gilbert, 2020). Unfortunately, it is well-known that sequential allocation is not strategyproof. This means that agents may try to manipulate the mechanism by reporting untruthful preferences (Aziz et al., 2017a). As a consequence, the strategic dimension needs to be addressed, and the study of Nash Equilibrium (NE) for this problem received particular attention. Aziz et al. (2017b) modeled the problem as a one-shot game and designed a linear-time algorithm to compute a pure NE. For the case of two players, (Levine and Stange, 2012) showed how agents arrive at the equilibrium by figuring out their opponent's last move first and reasoning backward. When the problem is modeled as a finite repeated game and perfect information, Kalinowski et al. (2013b) showed that the unique subgame perfect NE can be computed in linear time for two players. However, they show that computing one of the possibly exponentially many equilibria is PSPACE-hard one considering more agents.

In this paper, we propose a solution for the parking problem based on the sequential allocation mechanism. We take advantage of the particularities of the setting (e.g. the time constraints and the agents' priorities) to provide an algorithm that finds a Nash equilibrium on quadratic time.

3 PARKING PROBLEM

In this section, we start by introducing the *Parking Game Structure* model, (*PGS*, for short), which is the basis for defining and analyzing our proposed algorithm for addressing the parking problem.

3.1 Model

The PGS model describes the agents (or *players*), which represent the cars, as well as their needs and constraints. Also, the PGS takes into account all the specifications of the slots, in particular their location, their availability, the time they require to be reached from each entrance, and so on.

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

Definition 1 (Parking Game Structure)**.** The Parking Game Structure (PGS) is a tuple:

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

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*,
- $F = \{f_1, ..., f_n \mid f_i \in [0,1], f_i \neq f_j \text{ for } i \neq j \text{ and } 1 \leq i, j \leq n\}$ is a set of *resilience* values, representing how long the agents can wait for parking,
- $G = \{g_1, ..., g_l\}$ is a set of *gates*,
- $g : Agt \rightarrow G$ 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 3.2, 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 $Str : Agt \rightarrow S$. A *strategy profile* is an n-uple $\overline{s} = (s_1, ..., s_n)$ of strategies, one for each player. Formally, in $\overline{s}$, for each i, we have $Str(a_i) = s_i$. 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 $\overline{c}(\overline{s}) = (c_1(\overline{s}), ..., c_n(\overline{s}))$. Then, a *payoff* π of a strategy profile $\overline{s}$ is defined as a sum of all such $c_i(\overline{s})$, i.e., $\pi(\overline{s}) = \sum_i c_i(\overline{s})$, and by π_i we denote the $i-$th cost value of that tuple. We denote $\overline{o}$ for a tuple $(o_1, ..., o_n)$ of size n. For an agent $a_i \in Agt$, we let o_i be agent a_i's component in $\overline{o}$ and $\overline{o}_{-i} = (o_j)_{j \neq i}$.

Definition 2. Let $a_i \in Agt$ be an agent, $h = g(a_i)$ and $\overline{s} = (s_1, ..., s_n)$ be a strategy profile, with $s_i = Str(a_i)$. We define the *costs associated* to $\overline{s}$ as the tuple $\overline{c} = (c_1(\overline{s}), ..., c_n(\overline{s}))$ where each $c_i(\overline{s})$ is defined as follows:

$$c_i(\overline{s}) = \begin{cases} f_i \cdot (t_i - r_{(s_j,h)}) & \text{if (i) } (t_i - r_{(s_j,h)}) \geq 0, \text{ and} \\ & \text{(ii) there is no } a_{k \neq i} \text{ s. t.} \\ & f_k < f_i, \overline{s_k} = \overline{s_i}, \text{ and} \\ & (t_k - r_{(s_j,p)}) \geq 0, \\ & \text{with } g(a_k) = p \\ \infty & \text{otherwise} \end{cases}$$

In words, the value $c_i(\overline{s})$ 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 slot s_j.

Nash Equilibrium. Solution concepts are at the core of strategic reasoning and Game Theory because they are used to reason about the collective behavior of the agents. A well-conceived solution concept that ensures a robust form of satisfaction among players is *Nash equilibrium* (NE). 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 multiplayer game, all players, moving concurrently, reach a Nash equilibrium if none of them has the incentive to unilaterally deviate from that equilibrium. Formally, a strategy profile $\overline{s} = (s_1, ..., s_n)$ is a NE if, for each agent $a_i \in Agt$ and each alternative strategy $s_i' \in Str(a_i)$ we have

$$c_i(\overline{s}) \leq c_i(s_i', \overline{s}_{-i})$$

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 minimizing[2] the corresponding vector of associated costs $\overline{c}(\overline{s})$. 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 solution that provides, by definition, a satisfactory solution and, along with our setting, it just requires quadratic time. In the sequel, we aim to present the intuition of our proposed solution. We introduce our application scenario, the parking of a large hospital. By means of a toy example in this scenario, we describe how we pick the solution that is a Nash equilibrium. We also illustrate how this solution over-performs the greedy behavior of the players, in which each car takes the first available parking slot that satisfies their constraints.

[2]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.

3.2 Application Scenario

As a case study, we have focused on the parking area of the *Federico II Hospital Company* in Naples, one of the biggest and most specialized hospitals in the South of Italy, whose construction goes back to the early Sixties. The hospital is made of 21 building blocks, distributed over $440000m^2$. 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 pedestrians. The car gates are preceded by a road where cars line up for the necessary checks. On average, it is estimated that there are 4600 car accesses per day. There is no policy about the allocation of parking places and, except for a few reserved ones, each driver chooses their own slot. This disorganized solution produces huge traffic congestion, bottlenecks at the entrance, and an unbalanced distribution of 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, on 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 run-time. Also, we make use of all information the car passengers have to communicate 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, video cameras, etc.).

Having all this information at its disposal, the im-

plemented tool works as follows: it takes all cars in the queue on the roads in front of 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, the date and time of their appointments, possible walking limitations, and handicaps, etc. Note that the tool operates in stages, processing one bunch of cars at a time, as they are in the 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.

3.2.1 Running Example

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 approach and compare it with our solution. For a matter of presentation, we will recall the notion of Nash equilibrium.

Before proceeding, it is worth noting that at each instance of the game we consider, each car can enter the parking space through just one entrance. This means that we can get rid of g and G when dealing with a PGS, as well as the second index of the reaching-time values in RT. This also allows us using a simplified version of the definition of costs associated with strategy profiles. In other words, while the set RT provides $m \cdot l$ possible reaching values in general as stated in Definition 1 (with m the number of slots and l the number of gates), each instance of the game just requires dealing with RT as a vector of m values, i.e., $RT = \{r_1, ..., r_m\}$, where each r_i represents the time needed to reach the parking slot s_i from the physical gate through which the car is entering. When providing our solution to PSSG in Algorithm 1, we strongly rely on this observation, which leads to a natural reformulation of the model right after the vehicles are associated to the gates. Notably, we prefer to keep our PSG model as general as possible in order to accommodate other questions that require dealing with not *a priori* fixed entrances associated to cars. For example, it may be useful when devising an algorithm that also suggests in advance to a driver the gate to take. This, however, is not the target of this paper.

Example 1 (3x3-parking problem). 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, the second, and the third slot require 2, 3, and 4 minutes to be reached, respectively. We call such a game the 3x3-parking problem and it is reported in Figure 1.

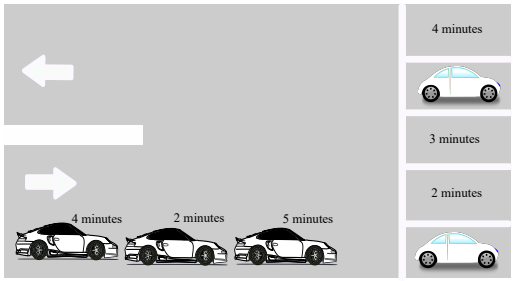

Figure 1: 3x3-Parking problem.

4 PRIORITY-BASED PARKING SELECTION

Following the model definition given in Definition 1 and the observations made above, we formally introduce the *Parking Slot Selection Game* as follows:

Definition 3 (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 1.
- Output: a strategic profile $(s_1, ..., s_n)$ such that it is a Nash equilibrium for $\mathcal{G}$.

In words, the PSSG looks for a strategy profile in which, with respect to the associated costs, no player has an incentive to unilaterally change his choice.

Similarly to the PSGG, one can define the *Greedy Parking Game* (*GPG*, for short). To give some details, first assume that in an GPG players are ordered, then the strategy profile $(s_1, ..., s_n)$ is such that for each agent a_i, it holds that s_i is the best choice (in terms of minutes to reach it) over $S \setminus \{s_1, ..., s_{i-1}\}$.

4.1 Algorithm

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

With the first iteration, the car with the lowest resilience index, *actualCar*, is selected from the queue,

Algorithm 1: Algorithm for the solution of the Parking Game Structure.

```
1: repeat
2:    actualCar ← priorityCar(carQueue).
3:    outcome ← ∞.
4:    for slot ∈ setAvailableSlots do
5:       po ← c(actualCar, slot).
6:       if po ≥ 0 and po < outcome then
7:          outcome ← po.
8:          strategy ← assignSlot(actualCar, slot)
9:          setNotAvailable(slot).
10:      end if
11:   end for
12: until carQueue ≠ null
13: return strategy
```

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 condition $po \geq 0$ capture the fact that the slot assignment must meet the cars' time restriction. The variable cost *outcome* is associated with 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(·)*.

4.1.1 Solution to the Running Example

Let us consider again the 3x3-parking problem described in Section 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, contrary to what we have seen with the greedy 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 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 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 $\bar{s} = (slot_2, slot_1, slot_3)$, with $\bar{c} = (1, 0, 0)$ (in bold in Table 1), and $\pi(\bar{s}) = 1$.

4.2 Game Theoretic Analysis

We now analyze our algorithm to show that it returns a strategic profile (parking slot assignment) in which no player wants to change his slot unless some other players want to change theirs.

Despite a pure Nash Equilibrium might not exist for any game, there are some special cases in which it does. Precisely, in Rosenthal (1973) it is shown that a pure Nash Equilibrium always exists when the payoffs are a non-decreasing function for each player. In our PSSG, the payoff associated with each player in a given slot remains constant when the other players change their strategies. Hence, we can use the results of Rosenthal (1973) to conclude that our game always admits a Nash Equilibrium. Furthermore, the proposed solution finds a strategy profile that is a Nash equilibrium for the parking problem.

Theorem 1. *The strategy profile returned Algorithm 1 is a Nash Equilibrium in PSSG.*

Proof (Sketch). Assume by contradiction that $\bar{s} = (s_1, ..., s_n)$ 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 $\bar{s} = (s_1, .., s_n)$ is the solution provided. □

4.3 Complexity Analysis

We now analyze the complexity of Algorithm 1.

Theorem 2. *The complexity of Algorithm 1 is quadratic with respect to the number of agents involved in the game, in the worst case.*

Table 1: Cost function values for 3-drivers-3-slots instance of the game. The table is read as follows: let T denote Table 1 and let the indexes i, j, and k be the strategy played by car_1 (in blue), car_2 (in red), and car_3 (in green), respectively. The position $T[i,j,k] = a,b,c$ represents the situation in which car_1, car_2, and car_3 are assigned to the slots indexed by i, j, and k respectively, with costs a, b, and c.

		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**	$1,\infty,0$	$1,\infty,0$
	$slot_3$	$0.5,\infty,0.018$	$0.5,\infty,0.018$	$0.5,\infty,0.018$	$0.5,0,0.009$	$0.5,\infty,0.009$	$0.5,\infty,0.009$	$\infty,0,0$	$\infty,\infty,0$	$\infty,\infty,0$

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 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 $|S|$ times, where S is the set of slots, according to Definition 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. As $|Agt| = n$ (Definition 1), we have $\mathcal{C}(1) = \sum_{j=1}^{n} \mathcal{C}(3) = \sum_{j=1}^{n} O(k) = O(nm)$.

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

5 EXPERIMENTAL RESULTS

In this section, we provide an experimental evaluation of the proposed algorithm.

5.1 Priority-Based vs Greedy Selection

We start by comparing the performances between executing a greedy solution to solve GPGs and Algorithm 1 to solve PSGGs. We first describe the greedy algorithm used in this benchmark.

Naïve Solution with Greedy Selection. When a car is approaching to the parking, a greedy 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 again the scenario described in Section 1. Assuming the agents use greedy selection 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.

We have considered 10 instances of problems which are with a growing number of cars and slots. For each experiment, the sets of values in the model (i.e., resilience values, agent-time, and reaching-time) were 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 Algorithm 1 is never worse than the greedy one. Moreover, by extending the experiment over 100 and 200 executions, our approach is strictly better than the greedy 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.

5.2 Benchmark in the Application Scenario

We have analyzed the behavior of Algorithm 1 in the management of a growing number of cars waiting for a parking slot, with respect to a fixed number of parking slots. All experiments have been executed on an Intel®Core™i5-7300HQ CPU processor of 2.50 GHz, with 8 Gb RAM capacity. We have considered two scenarios and reported the corresponding

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

	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
PSSG	3/3	3/4	5/5	6/6	7/7	8/8	7/9	8/10	9/11	12/12
GPG	2/3	3/4	5/5	5/6	6/7	6/8	6/9	7/10	8/11	10/12

Table 3: Execution time (in seconds) of Algorithm 1 varying the number of slots and cars.

	200 cars	400 cars	800 cars	1600 cars	3200 cars	6400 cars	12800 cars	25600 cars	51200 cars
4600 slots	0.001	0.002	0.004	0.009	0.027	0.402	1,389	3.415	10.165
20000 slots	0.003	0.006	0.013	0.026	0.060	0.150	0.430	5.687	23.597

benchmarks in Table 3. The first one considers 4600 slots. Such a number is not picked at random, but it refers to the number of slots available inside the structure of our case study, including some private parking slots close by. The second one considers 20000 slots. This number was chosen because it corresponds to the number of available slots in the biggest parking space of the world (West Edmonton Mall in Canada).

To show the scalability of our algorithm, 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.

6 CONCLUSIONS

The parking problem is one of the most challenging questions in the automotive research field. Inspired by the intrinsic competitive nature of the problem, in which drivers compete among themselves in order to get a suitable parking slot, in this paper, we explored a game-theoretic perspective. Precisely, following a real case study, we have formally introduced a multi-player game structure model and an algorithm based on sequential allocation adjusted to the parking problem. The game model includes time constraints, which denote how much time each car has available to park and how long it takes to park in a specific slot from its initial position at the gate. The solution found by the algorithm is a Nash equilibrium, which allows focusing not just on the best choice for a single car, but rather on one that guarantees no agent can improve their utility by a unilateral change of strategy. The proposed algorithm works in quadratic time.

As an application scenario, we consider 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 the back to early Sixties. Since there, no parking policy has been ever adopted: except for a few reserved slots, a car entering the area can park in any slot. This reflects in serious traffic congestion and inefficient use of the slots every day. Conversely, our approach provides, for the first time, a valid and promising solution. In order to put it into practice, we are currently working on a mobile client application to help drivers to park, from the assignment of the slot while approaching the gate, up to the moment they leave the car. Our work sets the stage for future improvements not only in health care services offered by the hospital under consideration but also in facilities from different contexts with similar problems. Experimental results show that (i) our solution improved the number of slots assigned with respect to greedy parking behavior, (ii) the algorithm is scalable and can handle a large number of slots and cars.

A recent line of work investigates the application of formal methods and strategic reasoning for the automated synthesis and verification of allocation mechanisms Mittelmann et al. (2022); Maubert et al. (2021). An interesting direction for future work is to apply such techniques to evaluate the solutions to parking slot allocation with strategic agents. Yet another direction is to formalize the parking problem as a linear program. A similar approach has been recently applied to allocation problems, such as online task assignment Dickerson et al. (2018).

ACKNOWLEDGMENTS

This research is supported by the PRIN project RIPER (No. 20203FFYLK), the JPMorgan AI Faculty Research Award “Resilience-based Generalized Planning and Strategic Reasoning”, and the EU ICT-48 2020 project TAILOR (No. 952215). We thank Giuseppe Calise for his help with the experiments and the earlier versions of this paper.

REFERENCES

Ayala, D., Wolfson, O., Xu, B., Dasgupta, B., and Lin, J. (2011). Parking slot assignment games. In *ACM-GIS*, pages 299–308.

Aziz, H., Bouveret, S., Lang, J., and Mackenzie, S. (2017a). Complexity of manipulating sequential allocation. In *AAAI*, pages 328–334.

Aziz, H., Goldberg, P., and Walsh, T. (2017b). Equilibria in sequential allocation. In *Proc. of ADT 2017*.

Aziz, H., Kalinowski, T., Walsh, T., and Xia, L. (2016). Welfare of sequential allocation mechanisms for indivisible goods. In *Proc. of ECAI 2016*, pages 787–794.

Aziz, H., Walsh, T., and Xia, L. (2015). Possible and necessary allocations via sequential mechanisms. In *Proceedings of IJCAI 2015*, pages 468–474.

Balzano, W., Murano, A., and Stranieri, S. (2017). Logic-based clustering approach for management and improvement of vanets. *J. High Speed Networks*, 23(3):225–236.

Balzano, W., Murano, A., and Vitale, F. (2016). V2V-EN - vehicle-2-vehicle elastic network. volume 98 of *Procedia Computer Science*, pages 497–502. Elsevier.

Balzano, W. and Stranieri, S. (2019). Acop: an algorithm based on ant colony optimization for parking slot detection. In *WAINA*, pages 833–840.

Belkhala, S., Benhadou, S., Boukhdir, K., and Medromi, H. (2019). Smart parking architecture based on multi agent system. *IJACSA*, 10:378–382.

Bouveret, S. and Lang, J. (2011). A general elicitation-free protocol for allocating indivisible goods. In *Proc. of IJCAI 2011*, pages 73–78.

Brams, S. J., Edelman, P. H., and Fishburn, P. C. (2003). Fair division of indivisible items. *Theory and Decision*, 55(2):147–180.

Brams, S. J. and Straffin Jr, P. D. (1979). Prisoners' dilemma and professional sports drafts. *The American Mathematical Monthly*, 86(2):80–88.

Budish, E. and Cantillon, E. (2007). Strategic behavior in multi-unit assignment problems: Theory and evidence from course allocations.

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J. A., Phelps, S., Rodríguez-Aguilar, J. A., and Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica (Slovenia)*, 30(1):3–31.

Dickerson, J. P., Sankararaman, K. A., Srinivasan, A., and Xu, P. (2018). Assigning tasks to workers based on historical data: Online task assignment with two-sided arrivals. In *Proc. of the 17th Int. Conf. on Autonomous Agents and MultiAgent Systems, AAMAS 2018*, pages 318–326.

Flammini, M. and Gilbert, H. (2020). Parameterized complexity of manipulating sequential allocation. In *Proc. of ECAI*, volume 325, pages 99–106.

Jioudi, B., Amari, A., Moutaouakkil, F., and Medromi, H. (2019). e-parking: Multi-agent smart parking platform for dynamic pricing and reservation sharing service. *IJACSA*, 10(11).

Kalinowski, T., Narodytska, N., and Walsh, T. (2013a). A social welfare optimal sequential allocation procedure. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 227–233.

Kalinowski, T., Narodytska, N., Walsh, T., and Xia, L. (2013b). Strategic behavior when allocating indivisible goods sequentially. In *Proc. of AAAI 2013*.

Kokolaki, E., Karaliopoulos, M., and Stavrakakis, I. (2013). On the efficiency of information-assisted search for parking space: A game-theoretic approach. In *IWSOS*.

Levine, L. and Stange, K. E. (2012). How to make the most of a shared meal: Plan the last bite first. *The American Mathematical Monthly*, 119(7):550–565.

Lin, T., Rivano, H., and Le Mouël, F. (2017). A survey of smart parking solutions. *IEEE Transactions on ITS*, 18(12):3229–3253.

Lu, X.-S., Guo, R.-Y., Huang, H.-J., Xu, X., and Chen, J. (2021). Equilibrium analysis of parking for integrated daily commuting. *Res. in Transportation Economics*.

Małecki, K. (2018). 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.

Maubert, B., Mittelmann, M., Murano, A., and Perrussel, L. (2021). Strategic reasoning in automated mechanism design. In *KR*, pages 487–496.

Mittelmann, M., Maubert, B., Murano, A., and Perrussel, L. (2022). Automated synthesis of mechanisms. In *IJCAI*, pages 426–432. ijcai.org.

Okoso, A., Otaki, K., and Nishi, T. (2019). Multi-agent path finding with priority for cooperative automated valet parking. In *ITSC*, pages 2135–2140.

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

Rad, F., Pazhokhzadeh, H., and Parvin, H. (2017). A smart hybrid system for parking space reservation in vanet. *JACET*.

Rosenthal, R. W. (1973). A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67.

Safi, Q. G. K., Luo, S., Pan, L., Liu, W., Hussain, R., and Bouk, S. H. (2018). Svps: Cloud-based smart vehicle parking system over ubiquitous vanets. *Computer Networks*, 138:18–30.

Senapati, B. R. and Khilar, P. M. (2020). Automatic parking service through vanet: A convenience application. In *ICCAN*, pages 151–159.

Shoup, D. (2005). The high cost of free parking.

Van Damme, E. (1991). *Stability and perfection of Nash equilibria*, volume 339.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John wiley & sons.