Parking Problem with Multiple Gates

Francesco Noviello[1], Munyque Mittelmann[1], Aniello Murano[1], and Silvia Stranieri[1]

University of Naples Federico II, Naples, Italy
francesco.noviello@studenti.unina.it
{aniello.murano, munyque.mittelmann, silvia.stranieri}@unina.it

Abstract. This work focuses on investigating parking problems with time constraints using a game-theoretic approach, specifically in a multi-gate scenario. The cars are treated as agents in a multi-player game where they compete for parking spots at entry gates that have no limit. We propose a priority-based algorithm for allocating parking spaces to address the problem. This algorithm guarantees a Nash equilibrium solution in quadratic time, which depends on the number of cars rather than on the number of gates. Additionally, we compare the performance of the proposed algorithm to a Greedy allocation method. The experimental results indicate the effectiveness of the proposed algorithm. Overall, the study highlights the potential of game-theoretic approaches for solving parking problems.

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

1 Introduction

The parking process is one of the serious social problems that daily involves our cities, where the demand for parking spaces often exceeds the available supply [14]. In this process, drivers compete against each other in order to get a parking slot, and parking problems in cities and urban areas are becoming increasingly important and have been one of the most discussed topics by both the general public and professionals. The difference between parking supply and demand has been considered the main reason for metropolis parking problems, and lacking it shows close relation with traffic congestion, traffic accidents, and environmental pollution. Although an efficient parking system can improve urban transportation, the city environment, and the quality of life for citizens, the problem is often overlooked in urban planning and transportation.

There are many causes for the deficit of available parking slots w.r.t. the demand, including the high activity concentration with a high rate of cars in the same area (such as commercial, medical, and governmental buildings) or miscalculation of parking demand expected. Conventionally, in urban planning, parking problems are solved with solutions based on different ideas, such as planning solutions, in which the number of parking spaces is calculated by urban

planners, or managed parks shared by buildings. Therefore, with the growth of Artificial Intelligence applications to automotive and the increased request for smart solutions to park, this problem is the inspiration for this work. In particular, we investigate the multi-gate parking problem with time constraints using a game-theoretic approach.

Related work The literature on smart parking solutions is extensive and varied. In [20], we 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 [5–8, 30–33]. Less common is the use of game-theoretic approaches to address the parking problem. An exception is [18], which is probably the closest to us, indeed we 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 [22], drivers' behavior is simulated by modeling the environment on the basis of cellular automata. In [9] 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, [15] provides an E-parking system, based on multi-agent systems aimed to optimize several users' preferences. In [28], the authors manage the parking problem with a cooperative multi-agent system, by relying on a priority mechanism. In [29], 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, [21] also considers the concept of Nash equilibrium applied to cars, but it is used to talk about traffic rather than parking.

Also related to our research are the problems of multi-agent resource allocation, which is a central matter when considering the distribution of resources amongst agents that can influence the choice of allocation [13]. In particular, the sequential allocation mechanism is a solution widely studied in the literature [3, 4, 10, 16, 17, 19] and has been considered in several real-life applications (for instance, to organize draft systems [11] and to allocate courses to students [12]). In this work, we propose a solution for the multi-gate 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.

In [26], the parking problem is analyzed as a competitive multiplayer game, where each car is an interacting agent with the goal of finding an available slot that meets its own constraints, the goal of the parking problem is to park as many cars as possible while satisfying their requirements, such as parking in bounded time, respecting one's resilience, and obtains a Nash Equilibrium for the game. But this work studies a scenario with one gate and proposes a solution for this case, not considering that parking areas (e.g., for hospitals, offices, and malls) have more gates than one. Thus, the multi-gate setting has not been considered.

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 [13].

Our contribution This study builds upon the work [26], by extending it to the multi-gate scenario. We employ a game-theoretic approach to model the multi-agent parking problem, which enables the analysis of strategic solutions, as demonstrated through the identification of Nash equilibrium. We then propose an algorithm based on agent resilience that can identify a Nash Equilibrium for parking slot allocation in quadratic time, which depends only on the number of cars and not on the number of gates. The performance of the algorithm is compared to a Greedy solution, and the results show that the Nash algorithm satisfies a higher number of parking requests and leads to higher social welfare, indicating greater agent satisfaction. Specifically, the Nash-based algorithm outperforms the Greedy solution when the number of cars is equal to the number of parking slots (in this study, 20000). Therefore, we conclude that the proposed algorithm is preferable for accommodating agents' demands.

Outline We start by formally introducing the parking problem in Section 2. In Section 3, 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 4 we present experimental results. Section 5 concludes the paper.

2 Parking problem

We begin this section by presenting the Parking Game Structure model, PGS for short, which forms the foundation for defining and examining our suggested method to solve the parking issue. Formally the Parking Game Structure is defined as follows:

Definition 1. *The Parking Game Structure (PGS) is a tuple* $\mathcal{G} = (G, Agt, S, F, T, R)$*, where:*

- $G = \{g_1, g_2, ..., g_n\}$ *is set of gates;*
- $Agt = \{Agt_k\}_{k \in G}$*, where* $Agt_k = \{a_1, a_2, ..., a_{l_k}\}$ *be the set of agents at the gate* $k \in G$ *(i.e., the* cars *waiting for parking at* k*), with* $\bigcap_{i=1}^{n} A_i = \emptyset$*. We let* $l_k = |Agt_k|$ *be the number of cars at the gate* k*;* $l_k = |Agt_k|$ *be the number of cars at the gate* k*;*
- $S = \{s_1, s_2, ..., s_m\}$ *is the set of parking slots;*
- $F = \{F_k\}_{k \in G}$*, where* $F_k = (f_1, f_2, ..., f_{l_k})$ *is the list of* resilience *values for the agents in* Agt_k*, with* $f_i \in [0, 1]$ *for each* $i \in Agt_k$*;*
- $T = \{T_k\}_{k \in G}$*, where* $T_k = (t_1, t_2, ..., t_{l_k})$ *is the list of* time limits *for the agents in* Agt_k*, where* $t_i \in \mathbb{N}$ *represents the time the agent* i *has available for parking starting from gate* g_k*;*

– $R = \{R_k\}_{k \in G}$, *where* $R_k = (r_1, r_2, ..., r_m)$ *is the list of* reaching-times *for the gate* k*, where* $r_i \in \mathbb{N}$ *represents the time needed to reach the parking slot* i *from gate* g_k*, for each* $i \in S$*.*

The resilience values for the agents have a twofold usage: first, they create an ordering system among the agents, which is essential in determining their prioritization; second, these indexes significantly impact the final preemption order, which can have a significant effect on the overall outcome. The intuition is that the higher the resilience the less the priority for the agent.

For the purpose of simplicity, we make the assumption that all the resilience indexes are unique, meaning $f_i \neq f_j, \forall 1 \leq i < j \leq n$. The indexes in the set F can either be manually set or automatically determined. In the case of agents, the resilience index represents their capability. Therefore, a lower index value indicates a higher priority.

A *strategy* for an agent involves choosing an appropriate slot. A *strategy profile* is a set of n strategies, one for each player, represented as an n-tuple $\bar{s} = (\bar{s}_1, ..., \bar{s}_n)$. It is important to note that it is possible for multiple players to choose the same strategy. Next, the *costs associated* with the strategy profile $\bar{s}$ will be defined as a tuple of costs, denoted as $\bar{c} = (\bar{c}_1, ..., \bar{c}_n)$. We let $B > 0$ be a constant value denoting the highest cost any agent may have for parking.

Definition 2. *Let* $a_i \in Agt$ *be an agent and* $\bar{s} = (\bar{s}_1, ..., \bar{s}_n)$ *be a strategy profile. The cost* $\bar{c} = (\bar{c}_1, ..., \bar{c}_n)$ *is such that:*

$$\bar{c}_i(\bar{s}) = \begin{cases} f_i(t_i - r_i);\ \textit{if}\ (i)(t_i - r_i) \geq 0\ \ \& \\ \qquad (ii)(\nexists k \neq i : f_k < f_i \wedge s_k = s_i \wedge (t_k - r_k) \geq 0) \\ B,\ \textit{otherwise} \end{cases}$$

The cost value c_i is considered finite if agent a_i has sufficient time to reach the parking slot s_i and the slot has not been occupied by another agent a_k with lower resilience ($f_k < f_i$). In this case, the finite value of c_i reflects the amount of time remaining for the agent after reaching the assigned slot, relative to the total amount of time available to him. On the other hand, if the cost value is assigned as highest, B, it represents the worst outcome for agent a_i, meaning that they were unable to park at slot s_i. The utility of agent i for the strategy profile $\bar{s}$ is

$$u_i(\bar{s}) = B - \bar{s}$$

That is, $u_i(\bar{s})$ is the difference between the highest cost B and her actual cost c_i given the strategies $\bar{s}$. Finally, the *social welfare* is the sum of utilities of among all agents in the system.

A strategy profile s is a Nash Equilibrium [27] if for all players i and each alternate strategy s'_i, we have that

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$$

In other words, no player i can change his chosen strategy from s_i to s'_i and thereby improve his utility, assuming that all other players stick the strategies

they have chosen in s. Observe that such a solution is self-enforcing in the sense that once the players are playing such a solution, it is in every player's best interest to stick to his or her strategy.

Then, the total cost, denoted as π, of a strategy $\bar{s}$ is defined as the sum of all the cost values in the tuple $\bar{c}$, that is $\pi(\bar{s}) = \sum_i c_i$. The i-th cost value in the tuple is represented as π_i.

Example 1. Let us now consider an example to illustrate the concepts and then formally introduce the PGS. In Figure 1, we consider a parking lot with 9 available slots, placed at various places in the park, and 9 cars looking to park, where t_i indicate agent i's available time to park, f_i indicates her resilience, and the values on slots indicate the reaching-time. For simplicity of the example, we assume all slots have the same reaching time from all gates.

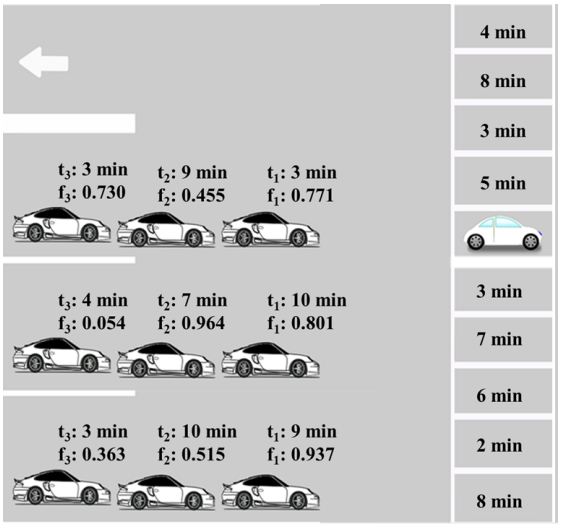

Fig. 1: Parking game with 3 gates, 9 cars waiting to park, and 9 available slots

3 Nash-based parking selection

We now describe the proposed algorithm for the parking slot selection game. Algorithm 1, called *nash*, first creates an ordered list of cars by iterating on each gate, appending in a local list *priorityQueue* the highest priority car of each gate. Then, it calls the assign algorithm, Algorithm 2, to be used on this ordered list.

This works by selecting the highest priority car, the first in the list, setting the outcome as B and, for each available slot evaluating the payoff with $c(\cdot)$, which evaluates cost according to Definition 2.

If the slot assignment must meet the cars' time restriction, the outcome value is updated and set as the best-computed cost, and the slot with the best result is assigned to the car. Once assigned, the slot is removed from the set of available ones.

Algorithm 1 *nash(carsQueue, availableSlots)*

Input: a vector of lists of cars *carsQueue*, a list of slots *availableSlots*
Output: a list of strategies (the strategic profile) *strategy*

```
1: repeat
2:    prirorityQueue ← ∅
3:    for i ← 1 to |carsQueue| do
4:       if carsQueue[i] ≠ ∅ then
5:          car ← priorityCar(carsQueue[i])
6:          priorityQueue ← append(prirorityQueue, car)
7:          carsQueue[i] ← remove(carsQueue[i], car)
8:       end if
9:    end for
10: until max(carsQueue) = 0
11: return assign(priorityQueue, availableSlots)
```

In the algorithms, $max(v)$ is a function that returns $\max_{1 \leq i \leq |v|}(|v[i]|)$, that is, the maximum size among the elements of the vector v. The function $append(a, b)$ (similarly $remove(a, b)$) returns the list a augmented with the element b (resp. obtained by removing b). $get(a, i)$ returns the i-th element of a list a.

Example 2. Let us recall the example introduced in Section 1. The cars would have been associated with the slots by the Greedy algorithm as demonstrated in Figure 3, resulting in parking for only 7 out of 9 cars. However as shown in Figure 3, parking for all cars in the queue is provided by the proposed algorithm.

In this example we see that the Nash equilibrium-based algorithm performed better in terms of social welfare, despite taking slightly longer to execute. Additionally, the Nash equilibrium-based algorithm was able to park the same number of cars as the greedy algorithm, mitigating any potential disadvantages of the longer computation time.

3.1 Algorithm Analysis

First, we show that Algorithm 1 always finds a strategy profile that is Nash equilibrium.

Theorem 1. *Algorithm 1 computes the Nash equilibrium for the game.*

Proof. Assume by contradiction that $\bar{s} = (\bar{s}_1, .., \bar{s}_n)$ is the solution provided from our algorithm and it is not a Nash equilibrium. Next, for the definition of Nash equilibrium, there must be an agent, say agent a_i, whose strategy s_j is not

Algorithm 2 *assign(carQueue, availableSlots)*

Input: a vector of ready cars *carQueue*, a list of slots *availableSlots*
Output: a list of strategies (the strategic profile) *strategy*

```
1: strategy ← ∅
2: while carQueue ≠ ∅ and availableSlots¬∅ do
3:    actualCar ← get(carQueue, 1)
4:    outcome ← B
5:    tmpSlot ← get(availableSlots, 1)
6:    for i ← 1 to |availableSlots| do
7:       slot ← get(availableSlots, i)
8:       po ← c(actualCar, slot)
9:       if po ≥ 0 and po < outcome then
10:         outcome ← po.
11:         tmpSlot ← slot
12:      end if
13:      strategy ← append(strategy, assignSlot(actualCar, tmpSlot))
14:      availableSlots ← remove(availableSlots, tmpSlot).
15:      carQueue ← remove(carQueue, actualCar)
16:   end for
17: end while
18: return strategy
```

optimal, with the strategies of the other players being fixed.
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 this exists, then it would be found during execution of the algorithm and it would be chosen as the final strategy for agent a_i. But this contradicts the hypothesis that $\bar{s} = (\bar{s_1}, .., \bar{s_n})$ is the solution provided, so this strategy is a Nash Equilibrium.

Now, let us evaluate the complexity of the algorithm.

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

Proof. Let us take into account the worst possible scenario, by considering the case in which no vehicle obtains a parking slot. For an arbitrary algorithm A, we denote as $C(A)$ as its computational complexity. The proof proceeds by analysing the complexity of the most expensive operations, from the inner ones to the outer ones.

In *nash* function, Algorithm 1, the ordering of cars is performed for each gate, so many times as $|G|$, on total number of cars, $|Agt|$. Assuming that $|G| = g$ and $|Agt| = n$, we can deduce that $C(nash) = O(g) \times O(n) + C(assign) = O(g \times n) + C(assign)$. In *assign* function, Algorithm 2, the loop for assignment of slot for each car is repeated many times as $|S| \times |Agt|$. Assuming that $|S| = m$ and $|Agt| = n$ and that in worst case m and n are of the same order, we can deduce that $C(assign) = O(m) \times O(n) = O(n) \times O(n) = O(n^2)$.

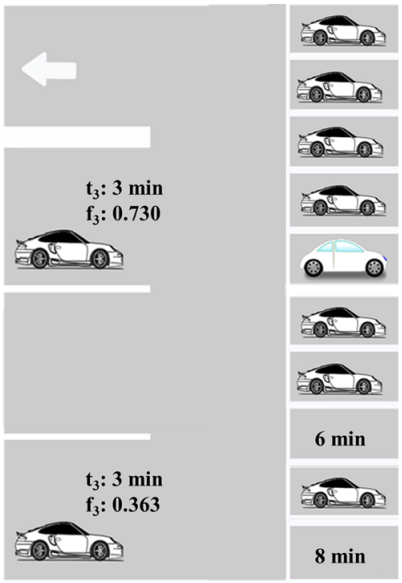

Fig. 2: Greedy solution for Example 1

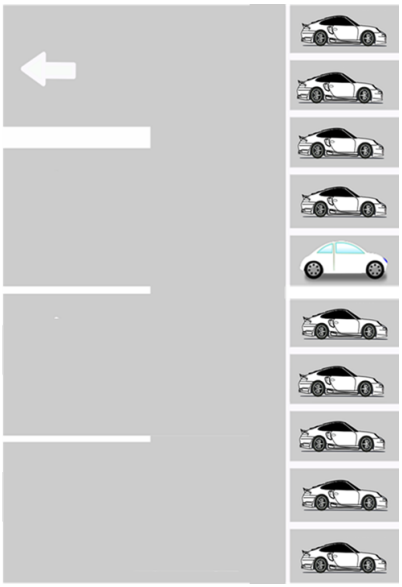

Fig. 3: Nash solution for Example 1

Given that, $C(nash) = O(g \times n) + C(assign) = O(g \times n) + O(n^2) = O(n^2)$. In other words, the algorithm's performance is proportional to the square of the input size (cars' number) and his growth factor will be influenced by the specific number of gates present but will maintain a quadratic nature.

4 Experimental results

In the experiments, each setting was executed 100 times. The results presented here are the average between those executions. All experiments have been executed on an AMD Ryzen™ 7 5700U with Radeon™ Graphics CPU processor of 1,80 GHz, with 16 Gb RAM capacity.

The algorithm proposed in the previous section was compared with a greedy selection of parking slots. In the Greedy algorithm, the cars in each gate park according to their original order in the queue and select the first slot matching their time requirements. This approach disregards the specific requirements and constraints of the other parked vehicles. Thus, it may not lead to an optimal allocation of parking spaces.

The experiments considered 20000 slots, a fixed number of agents, and a variable number of gates ranging from from 5 to 50 gates, as indicated on the x-axis of the graphs. In the graphs do not show Nash outperforming Greedy in terms of parked cars, except for case agents equal to slot, because this is consistent with previous results, which found that Nash results in either the same or more parked cars compared to Greedy. However, the Nash algorithm is

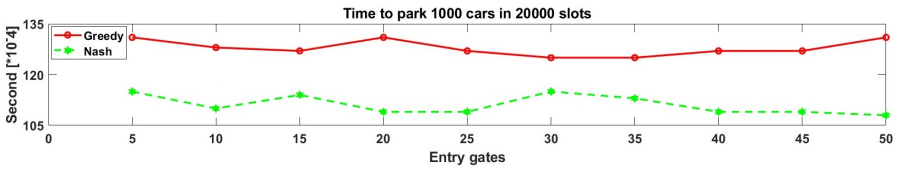

Fig. 4: Time to compute allocation for 1000 cars and 20000 slots

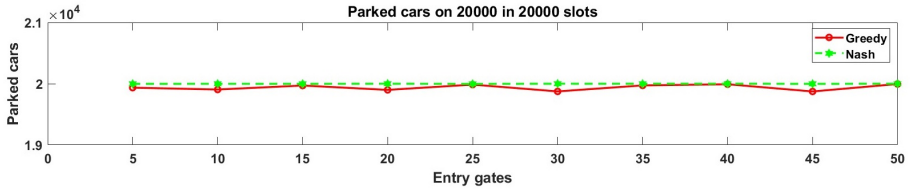

Fig. 5: Number of parked cars from an input of 20000 cars and 20000 slots

shown to park cars in a shorter time interval compared to the Greedy algorithm, demonstrating that the algorithm is highly efficient. Nonetheless, as the number of cars increases, so does the required time.

In the following, a comparison of the performance of two methods for solving GPGs and PSGGs, respectively, is presented. The greedy solution and the algorithm are executed 100 times, with random values for time limits and resilience. The experiments considered 20000 slots, a fixed number of agents, and a variable number of gates ranging, as indicated on the x-axis of the graphs below. It should be noted that each graph is labeled to provide the necessary context and avoid any ambiguity in interpreting the data. The presented results in Figure 4, indicate that the Nash approach outperforms the Greedy approach in terms of parking time, and takes hundreds of seconds to execute. Figure 5 presents the case in which the numbers are the same for cars and slots, we see that Nash algorithm outperforms the Greedy approach in terms of the number of successfully parked cars and of taken time, as in Figure 6, because Nash takes less than half a second whereas Greedy takes 0,65-0,70 s. In Figure 7, we can see that both algorithms park the same number of cars (that is, 20000 cars) in 1,5s on the average time for Nash and in 2-2,5s for Greedy. However, the Nash algorithm is shown to park cars in a shorter time interval compared to the Greedy algorithm, demonstrating that the algorithm is highly efficient.

In the same way as the number of parked cars was scored, we now display the results for social welfare, for the case where the number of agents and slots is equal to 20000. In Figure 8, it can be seen that the Nash algorithm obtained higher social welfare than the Greedy one.

The results demonstrate that when there are one thousand cars intending to park in twenty thousand slots, the Nash algorithm takes hundreds of seconds to execute. However, with larger numbers of cars, the Nash algorithm is again better in terms of the taken time, managing to park more cars in less time, although in

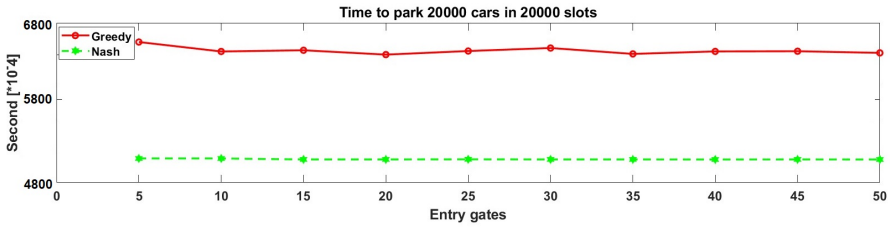

Fig. 6: Time to compute allocation for 20000 cars and 20000 slots

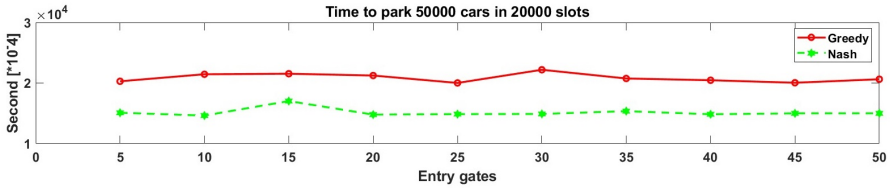

Fig. 7: Time to compute the allocation for 50000 cars and 20000 slots

seconds or tens. This makes it clear that the Nash algorithm is preferable to the Greedy algorithm, particularly in situations involving large numbers. Moreover, the time used in the configurations is proportional only to the total number of cars, not to the number of slots or gates present. In fact, as the number of gates varies, the time does not vary significantly.

5 Conclusion

This work presents a game-theoretic approach to formalize the multi-agent parking problem in a multi-gates scenario, allowing for the analysis of strategic solutions using Nash equilibrium. In particular, the proposed approach incorporates time constraints to represent slots' accessibility, and takes into account the agents' resilience in a competitive parking environment. We propose a multi-player game model that aims to solve the parking problem, and we develop an algorithm based on agent resilience and sequential allocation that aims to find a Nash equilibrium. This algorithm finds a Nash equilibrium for the allocation of parking slots in quadratic time and outperforms the Greedy solution in terms of the number of parking requests satisfied and social welfare. Moreover, the Nash algorithm is particularly effective when the number of cars is equal to the number of slots, making it preferable for accommodating agent demands, with performances significantly better. For future research, we aim to explore the use of formal methods and strategic reasoning to evaluate solutions for the parking problem [1, 2, 34]. Similar approaches have been considered in the context of synthesis and verification of allocation mechanisms [23–25].

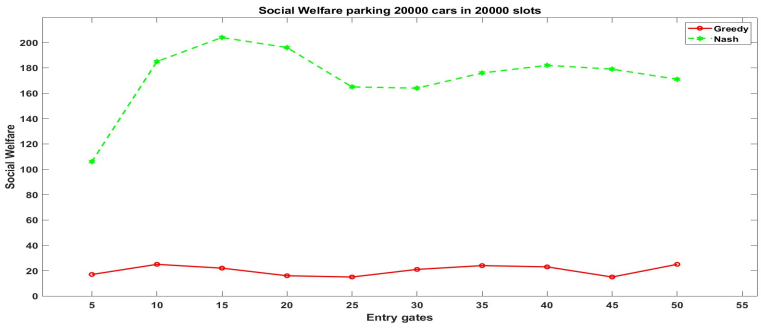

Fig. 8: Social Welfare obtained by the solution for 20000 cars and 20000 slots

Acknowledgments

This research is supported by the PRIN project RIPER (No. 20203FFYLK), the PNNR FAIR project, the InDAM project "Strategic Reasoning in Mechanism Design", and the EU ICT-48 2020 project TAILOR (No. 952215).

References

1. Aminof, B., Giacomo, G.D., Murano, A., Rubin, S.: Planning under LTL environment specifications. In: Benton, J., Lipovetzky, N., Onaindia, E., Smith, D.E., Srivastava, S. (eds.) Proc. of ICAPS. pp. 31–39. AAAI Press (2019)
2. Aminof, B., Murano, A., Rubin, S., Zuleger, F.: Verification of agent navigation in partially-known environments. Artif. Intell. **308**, 103724 (2022)
3. Aziz, H., Kalinowski, T., Walsh, T., Xia, L.: Welfare of sequential allocation mechanisms for indivisible goods. In: Proc. of ECAI 2016. pp. 787–794 (2016)
4. Aziz, H., Walsh, T., Xia, L.: Possible and necessary allocations via sequential mechanisms. In: Proc. of IJCAI 2015 (2015)
5. Balzano, W., Lapegna, M., Stranieri, S., Vitale, F.: Competitive-blockchain-based parking system with fairness constraints. Soft Comput. **26**(9), 4151–4162 (2022)
6. Balzano, W., Murano, A., Stranieri, S.: Logic-based clustering approach for management and improvement of VANETs. J. High Speed Networks **23**(3), 225–236 (2017)
7. Balzano, W., Murano, A., Vitale, F.: V2V-EN - vehicle-2-vehicle elastic network. Procedia Computer Science, vol. 98, pp. 497–502. Elsevier (2016)
8. Balzano, W., Stranieri, S.: ACOp: an algorithm based on ant colony optimization for parking slot detection. In: Proc. of AINA (2019)
9. Belkhala, S., Benhadou, S., Boukhdir, K., Medromi, H.: Smart parking architecture based on multi agent system. IJACSA **10**, 378–382 (2019)
10. Bouveret, S., Lang, J.: A general elicitation-free protocol for allocating indivisible goods. In: Proc. of IJCAI 2011 (2011)
11. Brams, S.J., Straffin Jr, P.D.: Prisoners' dilemma and professional sports drafts. The American Mathematical Monthly **86**(2), 80–88 (1979)

12. Budish, E., Cantillon, E.: Strategic behavior in multi-unit assignment problems: Theory and evidence from course allocations (2007)
13. Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J.A., Phelps, S., Rodríguez-Aguilar, J.A., Sousa, P.: Issues in multiagent resource allocation. Informatica (Slovenia) **30**(1), 3–31 (2006)
14. Ibrahim, H.: Car parking problem in urban areas, causes and solutions. In: 1st International Conference on Towards a Better Quality of Life (2017)
15. Jioudi, B., Amari, A., Moutaouakkil, F., Medromi, H.: e-parking: Multi-agent smart parking platform for dynamic pricing and reservation sharing service. IJACSA **10**(11) (2019)
16. Kalinowski, T., Nardoytska, N., Walsh, T.: A social welfare optimal sequential allocation procedure. arXiv preprint arXiv:1304.5892 (2013)
17. Kalinowski, T., Narodytska, N., Walsh, T., Xia, L.: Strategic behavior when allocating indivisible goods sequentially. In: Proc. of AAAI 2013 (2013)
18. Kokolaki, E., Karaliopoulos, M., Stavrakakis, I.: On the efficiency of information-assisted search for parking space: A game-theoretic approach. In: IWSOS (2013)
19. Levine, L., Stange, K.E.: How to make the most of a shared meal: plan the last bite first. The American Mathematical Monthly **119**(7), 550–565 (2012)
20. Lin, T., Rivano, H., Le Mouël, F.: A survey of smart parking solutions. IEEE Transactions on ITS **18**(12), 3229–3253 (2017)
21. Lu, X.S., Guo, R.Y., Huang, H.J., Xu, X., Chen, J.: Equilibrium analysis of parking for integrated daily commuting. Res. in Transportation Economics (2021)
22. Małecki, K.: A computer simulation of traffic flow with on-street parking and drivers' behaviour based on cellular automata and a multi-agent system. Journal of computational science **28**, 32–42 (2018)
23. Maubert, B., Mittelmann, M., Murano, A., Perrussel, L.: Strategic reasoning in automated mechanism design. In: Proc. of KR. pp. 487–496 (2021)
24. Mittelmann, M., Maubert, B., Murano, A., Perrussel, L.: Automated synthesis of mechanisms. In: Proc. of IJCAI. pp. 426–432. ijcai.org (2022)
25. Mittelmann, M., Maubert, B., Murano, A., Perrussel, L.: Formal verification of bayesian mechanisms. In: Proc. of AAAI. AAAI (2023)
26. Murano, A., Stranieri, S., Mittelmann, M.: Multi-agent parking problem with sequential allocation. In: Proc. of ICAART 2023 (2023)
27. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press (2007)
28. Okoso, A., Otaki, K., Nishi, T.: Multi-agent path finding with priority for cooperative automated valet parking. In: ITSC. pp. 2135–2140 (2019)
29. Pereda, M., Ozaita, J., Stavrakakis, I., Sanchez, A.: Competing for congestible goods: experimental evidence on parking choice. Scientific reports **10**(1) (2020)
30. Rad, F., Pazhokhzadeh, H., Parvin, H.: A smart hybrid system for parking space reservation in VANET (2017)
31. Safi, Q.G.K., Luo, S., Pan, L., Liu, W., Hussain, R., Bouk, S.H.: Svps: Cloud-based smart vehicle parking system over ubiquitous VANETs. Computer Networks **138**, 18–30 (2018)
32. Senapati, B.R., Khilar, P.M.: Automatic parking service through VANET: A convenience application. In: Progress in Computing, Analytics and Networking, pp. 151–159. Springer (2020)
33. Stranieri, S.: An indoor smart parking algorithm based on fingerprinting. Future Internet **14**(6), 185 (2022)
34. Yazdanpanah, V., Stein, S., Gerding, E.H., et al.: Multiagent strategic reasoning in the iov: A logic-based approach (2021)