A Parking System Based on Priority Scheme

Walter Balzano, Antonio Lanuto, Erasmo Prosciutto, Biagio Scotto di Covella, and Silvia Stranieri

Abstract The way the vehicles are placed in a parking space partitioned into parking slots reachable through internal routes means that the available space is not used at best of its capacity. In this paper, a solution model is proposed, adopting a chequered parking layout, which aims at optimizing the available surface and, consequently, increasing the capacity in terms of available parking slots. To this aim, an isomorphism with the game of fifteen is found and a solution is proposed.

Key words: Parking; Queue; Chequered parking; The game of fifteen

1 Introduction

With the considerable increasing of the number of cars on the roads, the problem of finding a parking space has increased [8, 7]. This not only affects the time wasted by motorists in finding parking spaces, but also affects other aspects. It has been seen that around 30% of urban traffic is also caused by cars looking for parking or parked badly, thus affecting air pollution. But if we also consider how the high demand for parking and the low availability of parking spaces has led, especially in large cities, to an exponential increasing of parking prices. The aim of this research is to find a way to optimise the available spaces in order to have more and more parking slots. This would also solve the other side problems explained above. Starting with a space at our disposal, the idea is to arrange the cars side by side, in a column,

University of Naples, Federico II
Balzano W. e-mail: wbalzano@unina.it · Lanuto.A e-mail: an.lanuto@studenti.unina.it · Prosciutto E. e-mail: e.prosciutto@studenti.unina.it · Scotto di Covella B. e-mail: b.scottodicovella@studenti.unina.it · Stranieri S. e-mail: silvia.stranieri@unina.it

with no movement routes but with a manoeuvring area to move the cars. In particular, the cars will be arranged in column order, so each car will have another car in front of it with a smaller remaining time. This allows us having always the cars that will free the parking space before the others near the exit. The manoeuvring area will serve us to ensure order within the car park. The following sections are divided as follows: in section 2 the related works are described; in section 3 an isomorphism with the fifteen puzzle game [6] is presented; in section 4 the adopted parking model and all its characteristics are explained; in section 5 the algorithm for managing chequered parking is described, and an example is also presented; in section 6 the final results of this research are presented; finally in section 7 the conclusions are explained.

2 Related Works

Smart parking, due to the growth of urban population and traffic congestion, is a crucial issue to address both in research and economically. Advances in technology allow drivers to more easily find parking spots through smart parking services. However, implementing a smart parking system is a complex and multidisciplinary process that requires extensive surveying and inspection. In [12], authors provide a smart parking guidance algorithm that supports drivers to find the most appropriate parking facility considering real-time status of parking facilities in a city. In [14], they present a feasible method to do parking planning, by transforming the problem into a kind of linear assignment problem and by taking vehicles as jobs and parking spaces as agents. They take distances between vehicles and parking spaces as costs for agents doing jobs, then they design an algorithm for this particular assignment problem and solve the parking planning problem. In [13], instead, authors provide a fingerprinting based solution for smart parking in indoor settings.

In this work we found an isomorphism with existing scenarios, in particular the fifteen game, similarly to what authors of [1] did by finding a solution to the parking problem by reducing it to the ant colony optimization problem.

The fifteen puzzle game has been largely studied in literature, and several solutions have been proposed [11, 5, 10], but it has never been applied to a parking scenario.

As far as we know, this is the first work designing a new parking environment with the aim of optimizing the available surface of the car park.

3 Fifteen Puzzle Isomorphism

In order to best describe the problem and the solution to be adopted, it was decided to resort to an isomorphism with a very famous puzzle which is *the game of fifteen*. The aim of the game is to reorder a matrix divided into four rows and four columns, which contains 15 tiles numbered from 1 to 15, using only one empty place in it. As can be seen, the game of 15 is based on the use of a matrix that can be represented by a very trivial chessboard $N * N$, and already here we begin to see a similarity with the proposed problem. In fact, the objective is to place the cars in a chessboard car park in such a way as to order them, from the bottom to the exit of the car park, in order of how long the car will remain in the car park. To do this, as in the game of fifteen, it is necessary to have a space where the parking spaces will be delineated, and an area that will represent the empty space of the game of fifteen. In this way, this area can be used as a temporary car park to facilitate the movement of cars within the car park itself.

4 Parking Model

It has been seen how *game of fifteen* represents a true isomorphism for checkerboard parking lots. Starting from the characteristics of the game, the goal is to demonstrate how, by using a buffer zone or cache zone, one can optimize the arrangement of cars in a checkerboard parking lot. To best clarify the basic idea of this paper, we will analyze the operation of the system in an ideal case. Starting from a parking lot having $n*n$ parking spaces, with a maneuvering area of n size and a checkerboard layout, we imagine that the parking lot is initially empty. When cars arrive at the parking lot, they tell the system how long they intend to stay inside it. The system, having taken this information, processes a priority queue in which at the top we will have the cars with a longer remaining time(Tr) than those following. Once this queue is processed, an arrangement of cars within the parking lot will be schematized, where at the bottom we will have the cars with a higher Tr. Importantly, it is not important to have a total sorting of the parking lot, but it is enough to have a sorting by column. In fact, in this way, the cars in front of everything, thus those at the top of each column of the matrix with which the parking lot is represented, will never have any obstacles to exit. Inside the parking lot we will have another queue that represents the actual presence of the cars in the parking lot, and again there is a temporal ordering of the cars, but in this case at the top we will have the cars with Tr lower than the others, and which will therefore exit the parking lot before the others The use of the buffer zone is necessary in case a car arrives with such a Tr that it cannot be parked in any free space, as it would not respect the ordering we set out to have. This situation makes it necessary to move

cars so that the latter arrives in the right place in relation to its *Tr*. Using the buffer zone, the cars that will have to make room for the newcomer will be moved to this zone, so that the correct place for the newly arrived car will be freed up. Having done this, the previously moved cars will be repositioned to the correct location without any problems. Note how using this *Tr*, each car always has a car in front of it that will start before it. Of course, in a real case it may happen that a user wants to pick up his car before the *Tr* expires, and even in this case the buffer zone can be made use of so that the car can be returned to the customer without any problems. So far we have talked about buffer zone and cache zone referring to the same parking area, and we have used the two terms interchangeably. Actually a difference is there even if it is the same parking area. The different designation given to this zone, refers to the use that is made of it. Specifically, when we speak of a buffer zone, we are referring to a temporary use of this zone to facilitate the movement of cars in the parking lot. When, on the other hand, we speak of a cache zone, we are referring to a different use. In fact, this zone could be used to put cars that are about to leave and free up spaces for new cars. Obviously, in this case, it must be a use that is not prolonged, since the cars stopped in this zone do not allow to move within the parking lot. We can therefore say that the proposed parking model can be divided into two main areas: (*i*) **Buffer/Cache Parking Area** - Part of the parking area used for moving cars or for temporary parking of exiting cars; (*ii*) **Effective Parking Area** - The main parking area, where cars will park in their assigned position according to the remaining time reported.

In addition to these two areas, one can consider a third parking access area, called **waiting area**, dedicated to the arrival of cars. It is here that the user, will communicate to the system the time he will stay and his car data (license plate) and then leave the car to those who will have to place it inside the parking lot.

5 Chequered Parking Algorithm

In this section, the he Chequered Parking Algorithm (CPA) is explained in its main steps and through a running example. The CPA algorithm can be divided into four working steps. It is taken for granted that the algorithm knows, from the first working stage, basic information such as the number of total parking spaces, those that are usable as parking spaces, and those that are reserved for the maneuvering zone or buffer zone.

- **Phase 1, Initialization**: the system asks the user for information about the incoming cars, such as the license plate number and the time he intends to stay(*Tr*) the car in the parking lot. The number of incoming cars accepted, of course, is equal to the number of vacancies, which at this stage will be equal to the number of total spaces in the parking lot. Once this

information is acquired, the algorithm creates a queue of incoming cars. Then this queue is sorted according to the *Tr* of each car, giving higher priority to cars with a higher *Tr*. Each car is saved in the form of a list of type [x,y] with x representing the car identifier and y the remaining time.
- **Phase 2, Parking Arrangement**: the system processes this priority queue and arranges the cars in the parking lot, according to a sorting by column. The cars will then be arranged so that each car has ahead of it either no car, or another car but with a lower *Tr* than it.

Having completed these first two steps, we enter the heart of the algorithm. In particular, we can say that the next steps represent dynamic parking management.

- **Phase 3, New Car Insertion**: this phase deals with the management of incoming new cars, implementing the same process as described above but taking into account the cars already in the parking lot. The algorithm, then, will manage the queue of incoming cars and look for the free place to place the incoming car. Of course, the vacancy may not coincide with the column sorting we are using. Starting then from the column of the free place found, the algorithm will check how many cars in that column need to be moved before the car can be placed, in order to comply with the sorting used. The algorithm then looks for the vacancy that allows the least number of cars to be moved.
- **Phase 4, Parking Rearrangement**: the algorithm now knows which column to work on and through a series of car moves, using the buffer zone as a maneuvering space, it manages to reposition them to the correct location while respecting a sorting by column.

5.1 Running Example

In order to better understand how it works, example images of how each algorithm presented above works are shown on the following pages.

Figure 1 shows an example of how the CPA algorithm works, specifically phases 1 and 2 are presented. Before going on to explain the tables, let us give some brief information. Specifically, the matrix represents a possible car park, within which, after trivial calculations, the two areas, presented above, were created with their respective parking spaces. In particular, the area *Buffer Parking Area* has at least as many spaces as there are in a column (in the event that there are columns with different sizes, the one with the most parking spaces is considered); empty spaces are marked *[0,0]*. Also to the left of the matrix is the initial queueEntry and the queueEntrySort, which is the queue we will be working on to arrange the cars in the car park. Finally, each time a car is placed in the car park, it is removed from the queueEntry and placed in the queueParking. For convenience, not all the steps are explained,

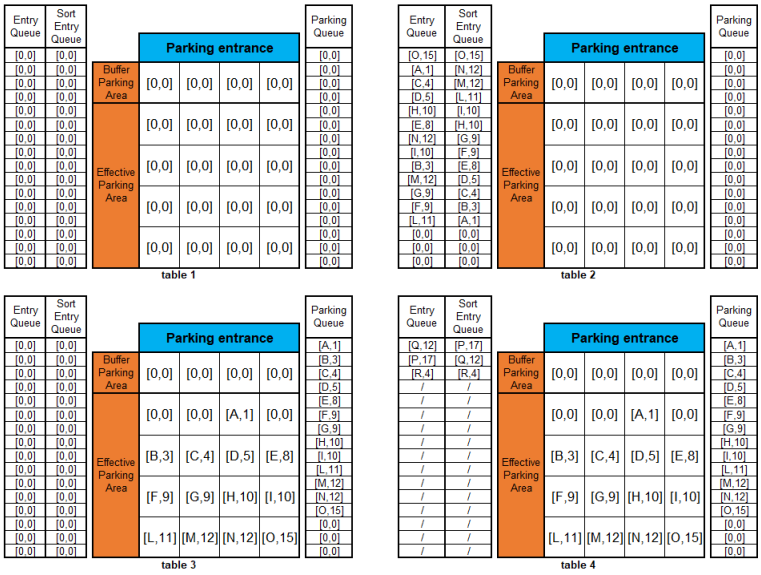

Fig. 1 CPA Algorithm: example Phase 1 and 2

but they are quite clear. In Table1 in 1, the matrix representing the car park is initialised at [0,0], so that it represents the empty car park. Then in table2 the queueEntry is filled and subsequently sorted so that it can be processed. In table3 the cars in the queueEntry are placed in their respective parking spaces, respecting the required column sorting. Finally in table4 we enter the heart of the algorithm with the arrival of new cars to park, but this time with the parking space not empty.

In figures 2 the CPA algorithm works in steps 3 and 4 presented above. In particular, we notice how the algorithm, once it has taken the incoming car, calculates the most efficient place, i.e. one that allows the least number of cars to be moved, where to park it. The free place, therefore, does not necessarily represent the place where the car should be parked, but represents the column where that place is and where we are going to move it. It is important to note that the place in the cache at the working column must always remain free to allow the car to be entered. As explained for figure 1, here too the cars are deleted from the queueEntry and inserted in the queueParking, always respecting an order of priority, in the case of the queueParking at the top we will always have the car with the lowest Tr. The insertion of the cars ['Q',12] and ['R',4] is not presented because it is superfluous, but would follow the same logic as the example in figure 2.

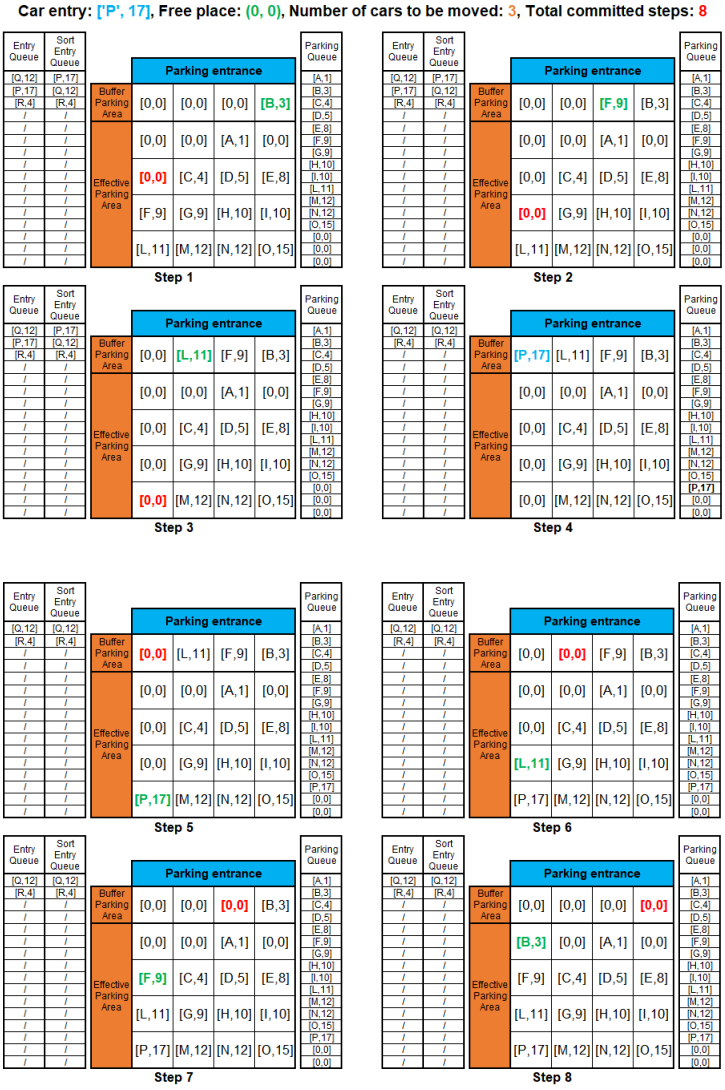

Fig. 2 CPA Algorithm: example Phase 3 and 4

6 Evaluation

In this section, we discuss the size of the Buffer that is needed for a fair compromise between space and time waste. We also provide a comparison of our solution with existing ones.

6.1 Buffer Size Evaluation

We now verify how the size for the buffer should be chosen. In particular, we start with an n*n matrix, and show that using the buffer area will only take k steps. Vice versa, we use a smaller and smaller buffer zone to show that with the latter we will employ at least k+1 steps. In this example we make use of a 3*3 matrix, with a buffer zone of size 3. We only demonstrate the worst case which is that the car park has only one free space and the incoming car must be placed in the last row of the matrix. Let us start by verifying how many steps it will take to sort this matrix n*n with the use of a buffer zone of size n. As one can see in figure 3, in order to place the car with Tr=10 in the right place, thus respecting the sorting by column, we need to move two cars and thus take a total of **6 steps**.

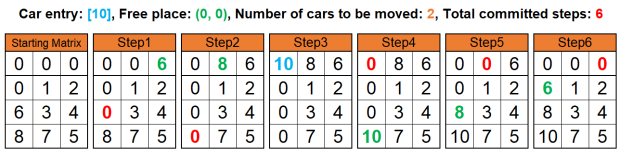

Fig. 3 BPA: example with CPA

Now let us check how many steps it will take to get from the *Starting Matrix* to the *Matrix Step6* using a buffer zone with fewer available parking places, in particular we first use a buffer zone of one place, then one of two places. Notice that, in a real scenario, in order for a car to be inserted in a column, the place corresponding to it in the buffer zone must remain available. It is important to note that in the counting of steps, neighbouring empty boxes can be considered as a single box. Furthermore, to explain these two cases, we use the logic of the game of 15 to move the cars around the board.

In the figure, we can see how using a BPA with only one parking space, a solution has not yet been reached after six steps. We can therefore already see how the BPA with only one parking space is less efficient than a BPA with three parking spaces used in the example in figure 3.

Also in this figure we can see how, using a BPA with only two parking spaces, after five steps we still have not arrived at a solution. We have stopped at step five because it is trivial to see how the step does not lead to the hoped-for solution, and it would also be very complex to show all possible combinations in the figure. We can, therefore, see how the BPA with two parking spaces is less efficient than a BPA with three parking spaces used in the example in figure three. These two simple examples prove that the buffer parking area must be at least as large as a column of the matrix.

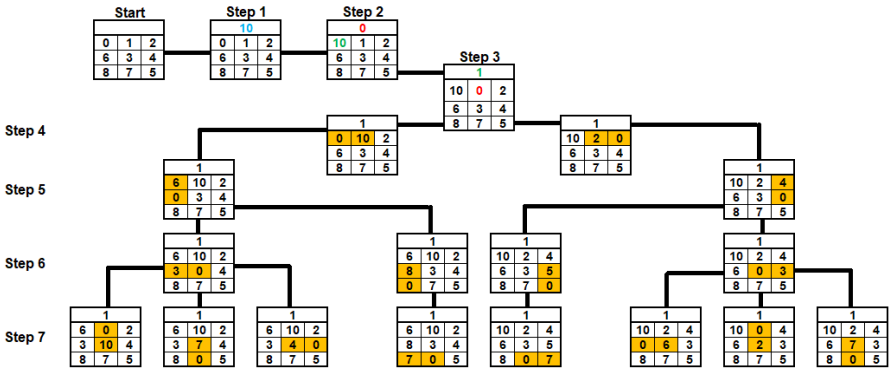

Fig. 4 BPA: example case 1

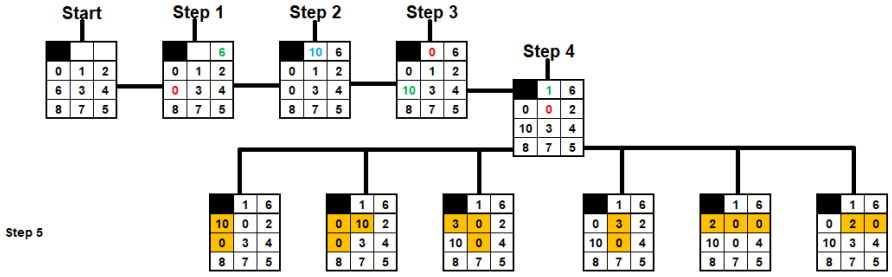

Fig. 5 BPA: example case 2

6.2 Comparison with existing algorithms

In this section we will show, with the help of an example, how chequerboard car parks, provided with a manoeuvring area, are an excellent solution not only for optimising movement within the car park itself, but also increase the capacity of available parking spaces and reduce wasted space. Let us start by constructing a chequerboard car park that has four parking spaces in each row and four in each column. With the algorithm used so far, we will then need an additional row with four parking spaces to create the Buffer Parking Area.

Under current regulations, a parking space must have an area of at least 2.5*5.0 metres (12.5 square metres). The chequered parking space to be designed must therefore have one side 10 metres long and another 25 metres long. As one can see in the figure, the car park has **16** parking spaces for the *Effective Parking Area*, and 4 parking spaces for the *Buffer Parking Area*. The total area occupied by the chequered parking area drawn above is 250 square metres (25*10m). With these dimensions we will see how a normal

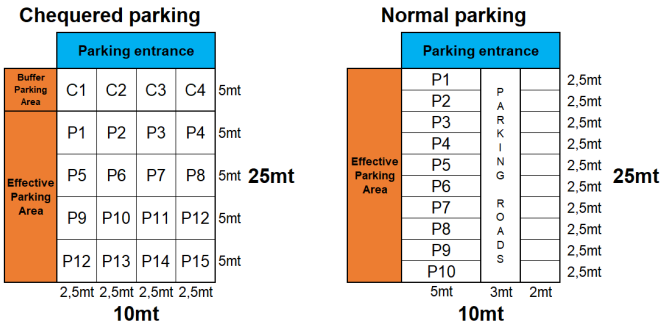

Fig. 6 Differences between two types of parking

car park, with even internal ways to move around, makes almost 50% fewer spaces available.

In fact, if we were to arrange the parking spaces in a single row on the long side, in order to comply with the regulations on the minimum dimensions of parking spaces and internal streets, we would not have more than 10 parking spaces available. Moreover, it can be seen that, in the subdivision of the space, a space of 2m*25m would remain, which would not allow the creation of new parking spaces and would therefore remain unused.

With this simple example we have shown how chequered parking spaces are a valid solution both for optimising space but also for having a higher actual parking capacity than normal.

7 Conclusions

In this work, an algorithm for smart vehicle allocation into parking slot is provided. Precisely, it is designed through the fifteen puzzle game. Indeed, the parking surface is designed as a chequered with some parking slots are reserved to move vehicle from one side to another (buffer).

Experimental results prove the effectiveness of this approach in terms of optimization of the parking surface. Indeed, it is compared with other algorithms relying on the standard parking surface division and the occupied space is seriously increased.

As hints for future development, we plan to study the parking problem from the perspective of a strategic reasoning approach through multi-agent systems [9], by possibly investigating learning techniques [2, 4, 3].

References

1. Marco Agizza, Walter Balzano, and Silvia Stranieri. An improved ant colony optimization based parking algorithm with graph coloring. In *International Conference on Advanced Information Networking and Applications*, pages 82–94. Springer, 2022.
2. Flora Amato, Luigi Coppolino, Giovanni Cozzolino, Giovanni Mazzeo, Francesco Moscato, and Roberto Nardone. Enhancing random forest classification with nlp in dameh: A system for data management in ehealth domain. *Neurocomputing*, 444:79–91, 2021.
3. Flora Amato, Luigi Coppolino, Francesco Mercaldo, Francesco Moscato, Roberto Nardone, and Antonella Santone. Can-bus attack detection with deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(8):5081–5090, 2021.
4. Flora Amato, Giovanni Cozzolino, Francesco Moscato, Vincenzo Moscato, and Fatos Xhafa. A model for verification and validation of law compliance of smart contracts in iot environment. *IEEE Transactions on Industrial Informatics*, 17(11):7752–7759, 2021.
5. Dler O Hasan, Aso M Aladdin, Hardi Sabah Talabani, Tarik Ahmed Rashid, and Seyedali Mirjalili. The fifteen puzzle—a new approach through hybridizing three heuristics methods. *Computers*, 12(1):11, 2023.
6. J Taylor Hollist. The fifteen puzzle. *The Mathematics Teacher*, 72(8):603–607, 1979.
7. MY Idna Idris, YY Leng, EM Tamil, NM Noor, Z Razak, et al. Car park system: A review of smart parking system and its technology. *Information Technology Journal*, 8(2):101–113, 2009.
8. Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3229–3253, 2017.
9. Vadim Malvone and Silvia Stranieri. Towards a model checking tool for strategy logic with simple goals. In *ICTCS*, pages 311–316, 2021.
10. Ben Morris and Anastasia Raymer. Mixing time of the fifteen puzzle. *Electronic Journal of Probability*, 22:1–29, 2017.
11. Charisma Tubagus Setyobudhi. Comparison of a* algorithm and greedy best search in searching fifteen puzzle solution. *International Journal of Computer and Information Technology (2279-0764)*, 11(3), 2022.
12. Jong-Ho Shin and Hong-Bae Jun. A study on smart parking guidance algorithm. *Transportation Research Part C: Emerging Technologies*, 44:299–317, 2014.
13. Silvia Stranieri. An indoor smart parking algorithm based on fingerprinting. *Future Internet*, 14(6):185, 2022.
14. Xuejian Zhao, Kui Zhao, and Feng Hai. An algorithm of parking planning for smart parking system. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 4965–4969. IEEE, 2014.