Application of Cuckoo Search Algorithm to Path Planning Problem of Automatic Navigation Parking Spaces in Parking Lots

Yu-Huei Cheng
Department of Information and Communication Engineering
Chaoyang University of Technology
Taichung, Taiwan
yhcheng@cyut.edu.tw

Cheng-Yao Kang
Department of Information and Communication Engineering
Chaoyang University of Technology
Taichung, Taiwan
s9518720@gmail.com

Che-Nan Kuo*
Department of Artificial Intelligence
CTBC Financial Management College
Tainan, Taiwan
cn.kuo@ctbc.edu.tw

***Abstract*—With the progression of urbanization, the problem of urban parking is becoming increasingly severe. Effective planning for parking spaces and vehicle route planning are crucial to resolving this issue. The aim of this study is to investigate the application of the Cuckoo Search Algorithm to the route planning for automatically navigated parking spaces in urban parking lots, with the goal of achieving more efficient and safe parking management. The Cuckoo Search Algorithm is a nature-inspired global optimization algorithm that has been proven to exhibit excellent performance in a multitude of complex optimization problems. We believe that by applying the Cuckoo Search Algorithm to the route planning for automatic navigation of parking spaces in urban parking lots, we can achieve more precise vehicle allocation, reduce parking time, and decrease the risk of vehicle collisions. To prove this, we have developed a new route planning method based on the Cuckoo Search Algorithm. This method takes into account factors such as the shortest route, the availability of parking spaces, the size and type of vehicles, and the location of other vehicles. Through a series of experiments, our method has demonstrated significant advantages over existing methods in terms of reducing parking time, decreasing parking distance, and minimizing the risk of collisions. Additionally, our method has exhibited excellent scalability when handling route planning problems in large urban parking lots.**

Keywords—Parking problems, Cuckoo Search Algorithm, Parking lots, path planning.

I. INTRODUCTION

The acceleration of global urbanization has led to issues with traffic congestion and parking, severely impacting the quality of life of city dwellers. Particularly in commercial and residential areas, the problems related to parking are especially prominent. Rational planning of parking lots is essential in relieving traffic pressure and enhancing the efficiency of travel. Faced with such challenges, the effective provision of optimal parking spaces and routes for drivers has become an urgent issue to resolve.

The Cuckoo Search Algorithm is a heuristic search algorithm inspired by the reproductive strategies of cuckoos. By simulating the egg parasitism behavior of cuckoos, this algorithm carries out a global search and exhibits a strong capacity for global search and high search precision. In a multitude of optimization problems, such as function optimization and project scheduling, the Cuckoo Search Algorithm has demonstrated superior performance. However, research regarding the application of the Cuckoo Search Algorithm to parking lot route planning remains relatively scarce.

Therefore, this study aims to explore the application of the Cuckoo Search Algorithm to the problem of route planning for automatically navigated parking spaces in parking lots. We apply the Cuckoo Search Algorithm to parking lots to ascertain the optimal parking spaces and the shortest routes to these spaces. Our method includes several steps: firstly, a parking lot model is established based on the layout of the parking lot and the initial position of the vehicle; then, the Cuckoo Search Algorithm is used to optimize the parking lot model to find the optimal parking spaces; finally, the shortest path algorithm is applied to determine the optimal route to the chosen parking space.

Through this research, we hope to offer a more effective solution to the current parking issues, further alleviate traffic congestion, and lay a solid foundation for the further application of automatically navigated parking spaces in parking lots.

II. METHIODS

A. Establishment Of Parking Path Planning Model

In parking lot modeling, we need to consider the structure, size, and features of the parking lot to effectively simulate real-world parking lots.

Step 1. Determine the size of the parking lot
Step 2. Set entrances, exits, and staircase exits
Step 3. Divide parking spaces and road networks
Step 4. Set floor levels
Step 5. Simulate parking lot occupancy
Step 6. Establish a visual simulation of the parking lot

For a more intuitive display of the parking lot simulation process, the parking lot status and vehicle routes can be presented through visualization methods. We create a GUI application using the tkinter module in Python, using different colors to represent empty parking spaces, occupied parking spaces, entrances, exits, and driving routes as shown in Fig. 1. In addition, a dropdown menu is created to select floors. The parking lot occupancy situation is randomly generated using the standard Python function “random”. Through these steps, a parking lot model is established and simulated. This helps analyze the impact of different parking strategies on vehicle travel time and distance, thus informing real-world parking lot design. We use the tkinter library to draw a visual interface for the parking lot, making it easy to view the planning results. By setting a “select” button, we can choose the floor level and view the parking lot layout of each floor.

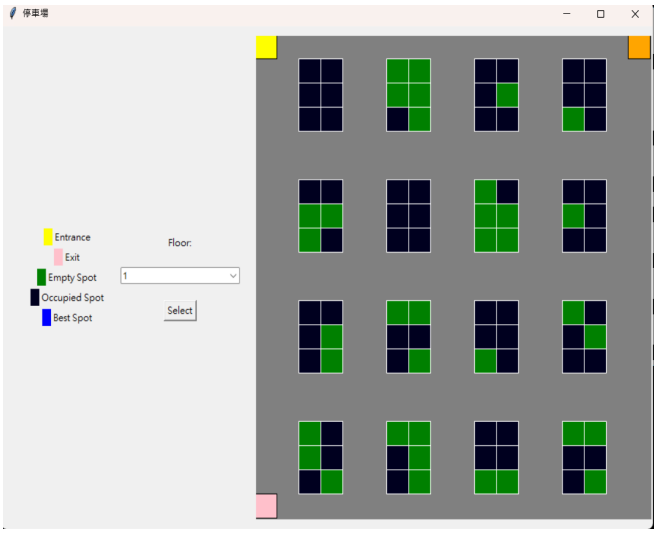

Fig 1. Multi-level parking lot created using tkinter.

B. Constraints

In this parking lot search problem, we must consider two primary constraints to ensure the feasibility of the obtained solution. The following provides detailed explanations of these two constraints:

1) No crossing over other parking spaces: When searching for a parking spot, it is essential to ensure that vehicles do not cross over other parking spaces. This requires careful consideration of the neighboring cells during the construction of our graphical representation. Only cells that can be directly accessed should be considered valid neighbors. This constraint ensures that the paths we generate adhere to the physical limitations present in reality.

2) Parking spaces must be entered and exited from the same direction: In order to facilitate smooth entry and exit of vehicles from parking spaces, it is necessary to consider the directions of entry and exit for each parking space. In this problem, we specify that vehicles must enter and exit the parking space in the same direction. To enforce this constraint, we ensure that only cells adjacent in the horizontal direction are considered valid neighbors when determining the optimal parking space. This approach prevents complex maneuvering of vehicles in confined spaces and ensures more realistic solutions.

In conclusion, when addressing parking lot problems, it is crucial to take into account these two constraints to ensure the practicality and realism of the resulting solution. By carefully considering these constraints during the construction of graphical representations and the application of search algorithms, we can identify the optimal parking spaces and shortest paths that adhere to these conditions.

C. Implementation of Cuckoo Search Algorithm

The Cuckoo Search (CS) algorithm is a novel nature-inspired optimization algorithm. This algorithm is mainly inspired by the unique parasitic reproductive strategy of cuckoos. Traditionally, cuckoos lay their eggs in the nests of other birds, which then incubate and take care of them. If the host bird discovers the cuckoo's egg, it may choose to discard or destroy it. If this happens, the cuckoo will seek a new nest to parasitize again. The Cuckoo Search algorithm has a wide range of applications in various practical problems, such as function optimization problems, engineering design problems, machine learning problems, and bioinformatics problems.

Here are the characteristics of the Cuckoo Search algorithm:

- Based on cuckoo's reproduction strategy: Cuckoos have a unique parasitic reproduction method, that is, they lay their own eggs in the nests of other birds and let these birds incubate their eggs. If the egg is found and discarded by the host bird, the cuckoo will find another nest and lay eggs again. The CS algorithm borrows this strategy, replacing the current solution with a new, better one.
- Lévy flights: The CS algorithm uses Lévy flights to update solutions, a random search method that makes it more likely to escape from local optima and perform global searches.
- Few control parameters: The CS algorithm only has two control parameters, making it relatively simple to adjust and implement.
- High efficiency in global search capability: Since the CS algorithm combines global and local search, it shows strong global search capability on many problems.
- Less dependence on the initial solution: Compared to some other optimization algorithms, the CS algorithm has less dependence on the initial solution, making it more robust.

The steps to use the Cuckoo Search (CS) algorithm for parking space search and path planning are as follows:

1) Initialization: In the Cuckoo Search algorithm, we first create or initialize a group of cuckoos. Here, each cuckoo actually represents a possible solution, or in this case, it may represent a possible parking space configuration. The position of each cuckoo is randomly selected in the search space of the solution.

2) Evaluation: The evaluation process involves calculating the fitness of each cuckoo. Fitness is a value that represents the quality of a solution (or the position of a cuckoo). In the case of parking spaces, this fitness might be based on the time or distance needed to reach each parking space, or other related conditions. Here is the formula for fitness:

$$d_{12=(x_2-x_1)^2+(y_2-y_1)^2} \quad (1)$$

$$\text{fitness\_value} = 1 / d_{12} \quad (2)$$

Formula one uses the Euclidean distance formula to find the shortest distance between the target and the parking space. Formula two is the fitness formula in the Cuckoo Search (CS) algorithm, where a higher fitness is better, so we set the fitness as the reciprocal of the distance.

3) Cuckoo Update: This step attempts to improve the current cuckoo, that is, to try to find a better solution. Specifically, this step updates the position of the cuckoo through a random process called "Lévy Flight". This process generates a new position (i.e., a new solution), and then we calculate the fitness of this new position.

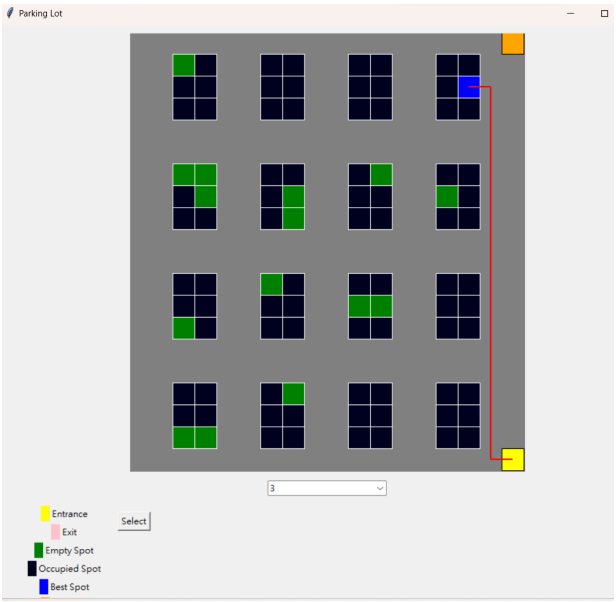

Fig 2. Optimal parking space and path search using Cuckoo Optimization Algorithm in Condition one.

4) Selection: In this step, we compare the new cuckoo (new solution) and the old cuckoo (old solution). If the fitness of the new cuckoo is better (i.e., the new solution is better), we replace the old cuckoo with the new one. If the old cuckoo is better, we keep the old one and ignore the new one.

5) Find the Optimal Solution: Compare the fitness values of all cuckoos to find the best cuckoo, i.e., the optimal parking space configuration and path planning.

6) Iteration: Repeat the above process multiple times. Each repetition continues until a stopping condition is reached. This stopping condition could be a fixed number of iterations or finding a solution that is good enough.

III. Results

Simulation experiments were conducted using python to apply the Cuckoo Search Algorithm proposed for the parking lot automatic navigation parking space path planning problem. The system configuration for the simulation experiment is shown in Table I.

TABLE I. CONFIGURATION OF THE SYSTEM

Type	Parameters
Operating system	Windows 11
CPU	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
Memory	16G
GPU	X
Compiler environment	Visual Studio Code
Development language	Python 3.7.7

Different parking lot layouts and conditions:

1) Condition one:

An 18*20 multi-story parking lot. The entrance is set at (0,0) on the first floor. The ramp exit is set at (0,19) on each floor. The ramp entrance is set at (17,19) on each floor, and the stair exit is set at (17,0) on each floor as shown in Fig. 2.

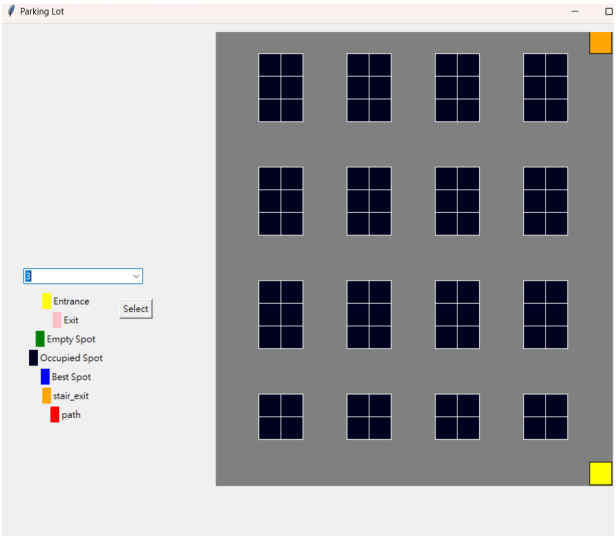

Fig 3. Optimal parking space and path search using Cuckoo Optimization Algorithm in Condition two.

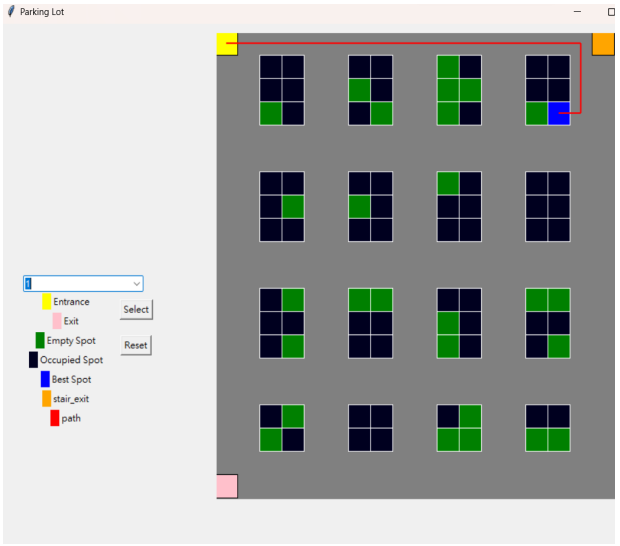

Fig 4. Optimal parking space and path search using Cuckoo Optimization Algorithm in Condition two.

2) Condition two:

An 18*20 multi-story parking lot. The entrance is set at (0,0) on the first floor. The ramp exit is set at (0,19) on each floor. The ramp entrance is set at (17,19) on each floor. The stair exit is set at (17,0) on each floor. However, the parking spaces on the second and third floors are full, leaving only the first floor. The schematic diagram are as shown in Fig. 3 and Fig. 4.

3) Condition three:

A large multi-story parking lot of 28*30. The entrance is set on the first floor at (0,0), the ramp exit is set on each floor at (0,29), the ramp entrance is set on each floor at (27,29), and the staircase exit is set on each floor at (27,0). However, the parking spaces on the third floor are full, and the parking spaces near the staircase on the second floor are also full. The schematic diagram are as shown in Fig. 5 and Fig. 6.

The experimental results show that the search time is usually longer in larger car parks. However, because the CS algorithm is often designed to be intelligent enough to handle car parks of different sizes and complexity, their time

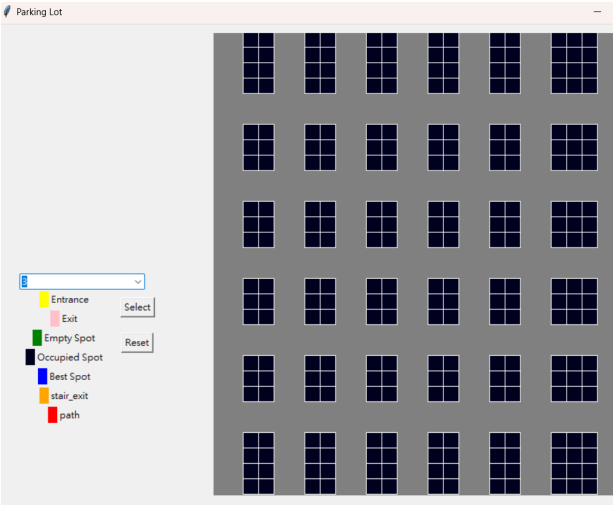

Fig 5. Optimal parking space and path search using Cuckoo Optimization Algorithm in Condition three.

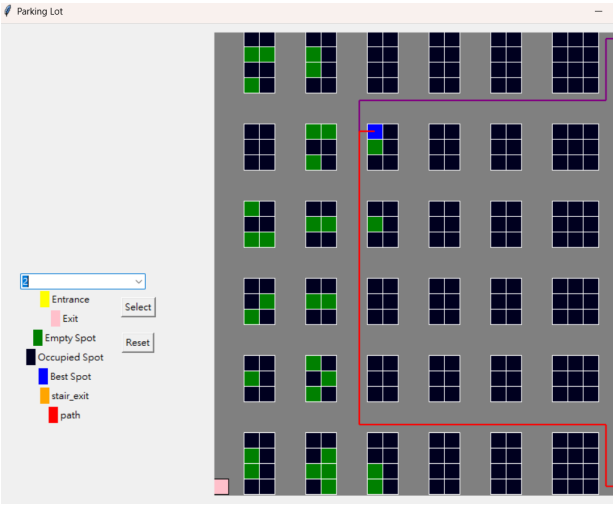

Fig 6. Optimal parking space and path search using Cuckoo Optimization Algorithm in Condition three.

consumption is not much different. Also, due to the factor of Lévy flights and the position of the defense is further from the staircase exit, it does not affect the time cost too much. The time cost comparison is shown in the Fig. 7.

IV. Conclusion

In this study, we have successfully applied the Cuckoo Search (CS) algorithm to solve the problem of finding the optimal parking space and planning the best arrival path in multi-storey car parks. This method demonstrates the effectiveness of the CS algorithm in solving highly complex combinatorial optimization problems. Moreover, this accomplishment provides a practical solution that helps drivers quickly and accurately find the optimal parking space and path in complex multi-story parking environments, thereby improving the overall efficiency of the car park and user experience.

However, despite the significant results we have achieved, there are still areas that need to be further improved and optimized. For example, our model may need to be further adjusted to adapt to car parks with special regulations or

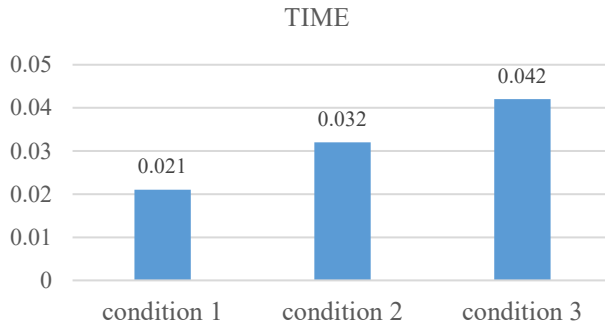

Fig 7. Comparison chart of time spent in three condition.

restrictions, or to special needs during peak periods. In addition, other factors in real car parks, such as the size and type of vehicles or the driving habits of drivers, which may affect the choice of parking spaces and the planning of routes, also need to be taken into account. In future research, we hope to further improve our model and algorithm to make them more in line with practical needs and situations.

Looking ahead, with the advancement of autonomous driving and intelligent transportation system technologies, we expect that the CS algorithm will have more application scenarios. For example, it can be integrated into the navigation system of autonomous vehicles to achieve automated optimal parking space search and path planning, further improving the efficiency and safety of vehicle operation. In addition, the combination of the CS algorithm with other advanced optimization methods, such as deep learning, is also a research direction worth exploring. We look forward to developing higher-quality and more efficient solutions in future research to meet the growing demand for intelligent transportation and inject new vitality into the development of modern urban transportation systems.

Acknowledgment

This work was partly supported by the National Science and Technology Council (NSTC) in Taiwan (under Grant no. 112-2221-E-432-002, 112-2218-E-005-010, 111-2622-E-324-004, and 111-2821-C-324-001-ES).

References

[1] R. Rajabioun, "Cuckoo optimization algorithm," *Applied soft computing,* vol. 11, no. 8, pp. 5508-5518, 2011.

[2] S. Jafari, O. Bozorg-Haddad, and X. Chu, "Cuckoo optimization algorithm (COA)," *Advanced optimization by nature-inspired algorithms,* pp. 39-49, 2018.

[3] Distance and similarity measurement methods. (2018, August 26). Taiwan Tribe. Access on https://www.twblogs.net/a/5b81ff832b71772165af1e64. Access date: 2023/08/31.

[4] Z. Liu, D. Li, Y. Yang, X. Chen, X. Lv, and X. Li, "Design and implementation of the optimization algorithm in the layout of parking lot guidance," *Wireless Communications and Mobile Computing,* vol. 2021, pp. 1-6, 2021.