Performance Comparison of Algorithms in Cake Cutting Game

Yensheng Ma[1], Shiyu Liu[2*]

[1]*School of Artificial Intelligence, Hebei University of Technology, Tianjin, 300132, China*
[2]*School of Cyber Engineering, Xidian University, Xi' an Shaanxi, 710126, China*
[*]*Corresponding author. Email: ariel_0818@163.com*

ABSTRACT
In recent years, games have become a hot field with great commercial value. And artificial intelligence in games became a important part of game design. In order to develop a sophisticated and efficient game AI, the algorithm implemented in the AI will play a role that cannot be ignored. Man-machine combat is a part of almost every game. However, how to choose the most appropriate algorithm to make computer players have better efficiency and winning rate, so as to bring real players a better experience is still a problem to be studied. Cake cutting is a simple game. We construct this game as an example to judge the efficiency comparison of four algorithms in this game, match different algorithms in pairs, and find the algorithm with the highest winning rate by considering factors such as first hand and then hand, the running time, memory occupation and CPU occupation rate of the game which are the factors for us to judge the efficiency. After research, we find that Minimax theorem is the algorithm with the best winning rate when taking no care of running time.

Keywords: *Game design; Greedy Algorithm; Dynamic Programming Algorithm; Depth first search (DFS); Minimax Theorem*

1. INTRODUCTION

1.1Background

Gaming is intimately related to the use of proxies and estimators in decision-making processes. Nowadays, video games and entertainment systems collectively make up the biggest industry in terms of turnover, more than music and cinema. [1] Because of this we can deduce that video games have become the preferred games of choice, exerting significant social and cultural influence over children, teens and adults [2].

It can be said that there is not much research on the four algorithms in the game field, and our cake cutting game can be extended to other aspects, which will bring high profit and efficiency for game industry.

1.2Subject

This paper mainly studies the application of four algorithms in cake cutting game.

Our cake cutting game means that we have a cut cake with randomly generated numbers on each piece of the cake. Players take turns to take a piece of cake, but only the cake on both sides of the one taken by the person in front can be took, until all the cakes are taken away. Finally, the player with the largest sum of cake numbers wins.

These four algorithms are greedy algorithm, dynamic programming algorithm, depth first search algorithm and maximum minimum theorem. We asked four computer players to apply four algorithms to choose which cake to fight against and choose the final winner among the four computer players.

Our judgment of the algorithm mainly depends on the winning rate of the game. When the same number of games are played, the algorithm applied by the winner has the advantage. In addition, we also discuss the efficiency of the algorithm. When winning the same game, consider the running time, memory occupation and CPU occupation of the game.

1.3What We Find

We mainly use the method of controlling variables to control all factors except judgment factors. And increase the number of confrontations to eliminate randomness. During the experiment, we mainly found mini-max has the highest success rate both first-hand and second-hand. The downside is that mini-max runs very long.

We assume this might be related to minimax's overall traversal and pessimistic thinking. The cake cutting game seems simple, but the specific idea can also be extended to other similar card games, which also has great commercial value.

2. FOUR ALGORITHMS

A cake-cutting game is essentially a game be- tween two players. Game refers to the process in which individuals or teams choose and implement behaviours or strategies that are allowed to be chosen at the same time or successively, and obtain response results or benefits from each other, depending on the information they have mastered under certain environmental conditions and rules [3].

Here are four algorithms that help players find ways to win in a more efficient way.

2.1 Greedy Algorithm

Greedy algorithm has no specific algorithm framework. The key of design is to make the best choice when solving the problem, that is, the local optimal solution in a sense.

Its basic idea is:

- Establish a mathematical model to describe the problem
- The problem is divided into several sub problems
- Each sub problem is solved to obtain the local optimal solution of the sub problem
- The local optimal solution of the sub problem is synthesized into a solution of the original problem

Because greedy algorithm selects the optimal strategy locally, and then combines all parts into a whole solution, the final result is not necessarily the overall optimal solution. In addition, because it is to find the local optimum, it also leads to its inability to find the global limit values, such as the maximum and minimum values. Because local constraints are defined in the process of solving, local constraints will also be attached to the final solution.

Also because of the special idea of local solution, the premise of greedy strategy is that the local optimal strategy can produce the global optimal solution. But in fact, the greedy algorithm is rarely applicable. It is limited when producing a possible solution for the problem of finding a mini- mum weight base in an independence system when the weights are taken from a finite range [4] . Generally, to analyse whether a problem is suitable for greedy algorithm, you can first select several actual data under the problem for special value analysis, and then make a judgment.

2.2 Dynamic Programming Algorithm

Dynamic programming is a mathematical method to solve the optimization of decision proces.

Dynamic programming is mainly used to solve the optimization problem with multiple repeated sub problems. It can decompose the original problem into similar sub problems, solve the solutions of sub problems and save them, so as to avoid solving the same problem repeatedly, and then gradually merge the solutions of non-repeated sub problems into the solutions of the original problem. How- ever, although this avoids repeated solution, it will occupy more space because the solution of the sub problem will be stored.

The problem solved by dynamic programming mainly has the following two characteristics:

- Overlapping Sub Problems: some sub problems will be calculated repeatedly
- Optimal Substructure: the optimal solution of the problem can be obtained from the optimal solution of a sub problem. Depth first search(DFS)

In introducing the DFS algorithm, we first introduce the decision tree structure.

A decision tree is a tree structure in which each case can be represented as a node. Like trees in nature, decision trees have root and leaf nodes. From the root node, the content of the root node represents the cake of the first selection. As we search for another cake gradually, we connect the node to be selected with the current node, indicating that we have searched for the second cake from the first one. Considering each case like this, a decision tree is formed.

Depth first search starts from the root node, finds the rightmost child node of the root node, and looks down gradually along the rightmost child node branch. When the rightmost leaf node is found, return the penultimate child node of the root node and continue to search downward. Re- peat this process step by step to the left until all nodes are traversed.

Algorithm idea:

1. Create an empty stack and an empty visit list.

2. Place the starting point and adjacent points in the stack and visit list in turn.
3. Take the last node in the stack out of the stack and obtain the adjacent points of the node from the graph
4. If the critical point has not been accessed, it is added to the stack and visit list
5. Output the node that is out of the stack
6. Repeat 3.4.5 until the stack is empty

DFS is a search algorithm, which needs to visit all nodes once, which is very time-consuming, so it is not suitable for the case of large amount of data.

When the traversal layer is very deep, there may also be insufficient storage space.

2.3 Minimax Theorem

Minimax is a very pessimistic algorithm. It always thinks that the opponent is strong, so it always chooses the algorithm with the least possible failure.

In addition to the second selection, each cake selection can only be made from two pieces at the gap, which results in only two cases for each node, representing two subtrees, which is also called a binary tree. Minimax algorithm is based on binary tree, traversing all nodes from leaf node to root node.

Combined with our cake-cutting game, mini- max describes the following scenarios:

Two people play the cake-cutting game, one named min and the other named max. Max al- ways wants his own cake score to be the largest, and Min always wants Max score to be the small- est. As Max, we always start with the leaf node layer. Starting from the bottom of the leaf node, the first layer of the binary tree is Max and the second layer is Min. Max is not selected as the bottom layer, followed by Min. In order to let Max select the smallest value, Min selects the smallest child node (the slice with smaller number) from the child nodes (all the slices) and fills it in its own nodes (It means this slice is Min's choice). Alternate to Max, and fill in the Max layer with the maximum value of the child node (It means this slice is Max's choice). This alternates until the root node is reached, at this time all the slices are taken.

Minimax essentially traverses all nodes. This method is simple and effective, and can also return better results, but there are many problems.

Very complex task situations: because mini-max algorithm contains the idea of" exhaustive", that is, access to all situations that should be accessed as much as possible. If the maximum depth of the tree is m and each point has b legal and effective action methods, the time complexity of the algorithm is O(bm)For real games, this shows that the running time will be very long and some are difficult to accept.

Minimax pessimism: because you always imagine that your opponent is strong, the purpose of each step is to reduce the loss as much as possible. This leads to the result of selection, which may give up better options because of" lack of courage".

3. EXPERIMENT

3.1 Combat methods

We realize the real cake cutting game through the python environment. In addition, four functions are completed to represent four algorithms. In the game, two players are set to call the corresponding function to represent the method they use to select the cake.

Because of the particularity of this game rule, the first player can choose from all the cake pieces when choosing to extract the cake pieces, while the second player can only choose two pieces at the edge of the gap when choosing again. The difference in the fast number of this kind of cake caused by choosing first and then shows that it needs to be divided into two cases for discussion.

3.2 Result presentation

For each case, we carried out 500 experiments. Table 1 shows the experiments' results. The first row in Table 1 represents the four algorithms that are performed first, and the first column represents the four algorithms that are performed later, so that we can see 16 situations in which the first one is engaged. The diagonal data represents the situation in which you are fighting against your- self.

The data for each cell indicates the probability of winning for the first hand. W = Number of wins for the first-hand algorithm, T = 500(Total number of experiments)

The calculation method is:

$$W \div T \tag{1}$$

Table 1: First Hand

2/1	**Greedy**	**DFS**	**DP**	**Minimax**
Greedy	74.7%	26.3%	28.2%	99.7%
DFS	92.6%	36.4%	76.8%	99.2%
DP	62.8%	26.2%	49.7%	99.4%
Minimax	34.7%	1.8%	1.2%	99.6%

3.3 Mutuality

When compared with the other three algorithms, minimax always has great advantages to win, and the winning rate is almost all over 99%. Secondly, the greedy algorithm performs better than DFS and DP algorithms. Compared with other algorithms (except minimax), greedy algorithm’s war victory rate is more than 60%. Followed by DP algorithm, the worst is DFS algorithm.

Minimax Wins So Much

Why did minimax show a huge advantage when it first started? This is related to their respective algorithm ideas.

When minimax selects the first cake, it will traverse all 32 pieces of the whole cake according to the algorithm principle described in 2.4. Turn the specific cake into a binary tree, select the leaf node, select the maximum value for yourself, and imagine that the other party will choose the mini- mum value, alternating layers until it traverses the root node. Finally, choose the best first cake that minimax thinks.

Minimax carefully considered the situation of the global game when choosing from the first piece. However, compared with the greedy algorithm, starting from the second selection, only local problems are considered in each selection without global consideration. The scope of each selection is only limited to the two slices at the optional gap, and the calculation is only limited to these two cakes without global consideration. So, the result is not the best.

Similar to greedy algorithm, DP algorithm divides the problem into subproblems. The algorithm stores the best results of all subproblems, and selects the results of subproblems as the best results that can represent the whole problem. How- ever, as a backhand, DP can only choose two cakes at the gap. Therefore, compared with minimax for all cake situations, DP is not comprehensive enough in each step.

As a backhand, DFS algorithm traverses all the cakes that can be selected at the beginning. However, in the search process, the search is only carried out in the fixed order of depth first, and the idea of defeating the other party is not taken into account in the selection process, which naturally leads to the disadvantage in the comparison of minimax with competitive ideas.

3.3.1dp and greedy algorithm

In the war between DP algorithm and greedy algorithm as backhand and DFS at the same time, we can also see that the winning rate of minimax is very close, both of which are 26%. At the same time, combined with the algorithm theory and experimental results, we speculate that the ability of DP and greedy algorithm is similar in the cake cutting game. However, when greedy and DP play DFS and minimax and DFS respectively, we can see from the winning rate (greedy minimax 34%, DP minimax 0%, greedy dfs 92%, DP DFS 76%) that DP is not as good as greedy algorithm.

The decision of each step of greedy algorithm is the optimal solution derived from the previous step, but all solutions before the previous step are discarded.

The decision of dynamic programming algorithm will contain a local optimal solution, which is not necessarily the optimal solution of the previous step. The cake cutting game is a continuous selection game. The choice of the previous step will greatly affect the next step, while the previous step will not have a great impact on the winning rate results brought by the choice of this step. Compared with greedy algorithm, this may lead to excessive consideration of worthless local solutions in DP decision-making.

3.4 Self-Combat

We can see from the data in the table that different algorithms have significantly different winning rates. Because both players use the same algorithm, it is only different when the first hand chooses the cake for the first time, which further explains the necessity of increasing the discussion of successive hands. If the winning rate is close to 50%, it shows that whether the first hand or the second hand has little effect on the winning rate. The more the winning rate deviates from 50% on both sides, the greater the influence of first hand and then hand.

Minimax has a huge advantage when it comes first. When the first cake is selected, the number of all cakes will be selected. Considering all the circumstances, it will have great advantages at the beginning. The specific reason is also similar to the great advantages of minimax against other algorithms discussed in 3.3.1 above. Here, even if minimax itself is the backhand, only calculating the remaining two pieces cannot make up for the great advantage of minimax’s first hand in traversing all cakes.

The results show that DP algorithm is different from other algorithms. He won 50% against himself, which shows that he is not affected by the first hand. When DP algorithm is compared with other algorithms, it can also be seen that whether DP is the first or the second hand, it has no great impact on the winning rate. DP first hand greedy algorithm: the winning rate is 28.2%; Grey algorithm of DP backhand Combat: the winning rate is 31.2% (1-62.8%); DP first hand DFS algorithm: the winning rate is 76.8%; DP backhand DFS algorithm: the winning rate is 73.8% (1-1.2%); DP first hand battle minimax algorithm: the winning rate is 1.2%; DP backhand battle minimax algorithm: the winning rate is 0.6% (1-99.4%)

The reason is still related to the calculation idea of DP algorithm. DP as the first hand, traverse all cake blocks for calculation, and finally find the most suitable one. When you get a piece of cake, you temporarily think it is the best cake. When you see the next piece of cake, compare it with the first piece. If the number of the second piece is larger, choose the second piece, otherwise continue to choose the first piece of cake. Re- peat this comparison until all the cakes are com- pared, and choose the best one as the first cake to choose. When DP the second choice, compare the two cakes that can be selected. Also choose a larger number. But because the difference between the numbers on the cake is not big, even if you get the largest cake first, you can't win the game directly. In the follow-up selection, if the backhand DP always chooses the cake with large number, it is entirely possible to make up for the digital difference brought by the first cake.

4. RUNNING TIME

In each case, we conducted 500 experiments. The first row in the table represents the four algorithms performed by the first hand, and the first column represents the four algorithms performed by the second hand, so we show 12 cases of first hand combat. The diagonal data indicates the situation of fighting with yourself.

Similar to the winning table, Table 2 shows the running time of 16 cases.

The result retains 3 decimal places. The unit is seconds(s)

Table 2: Running Time

2/1	Greedy	DFS	DP	Minimax
Greedy	0.067	0.900	0.474	3.612
DFS	0.849	1.145	0.885	4.148
DP	0.338	0.639	1.305	3.619
Minimax	1.009	1.333	0.892	0.098

From the table, we can see that the operaion time of minimax as a pioneer in the war with other algorithms (greedy algorithm, DFS and DP) is much longer than that of other algorithms. This is because all cake pieces need to be traversed when selecting the cake for the first time. Suppose that each cake is the first one and pushed to the end until all the cakes are selected.

Minimax takes a lot of time to traverse all situations, so it is not the best choice in some time limited games.

But we found that when minimax fought against itself, the time was unexpectedly short, as long as 0.098 seconds. Analysing the code, we think this may be because the two functions are the same, so they use the same decision tree. When two players' minimax traverse each cake, they assume that each other will have the best choice. Therefore, their decision-making is completely carried out according to the ideal decision tree established for the first time, so there will be no new decision tree construction and modification process, so the time consumption is greatly reduced.

5. MEMORY USAGE

5.1 Greedy Algorithm

Greedy algorithm as a backhand only needs to compare the two cakes at the optional gap. It is stipulated in the algorithm idea of greedy algorithm that it does not need to store complete data. Just compare the size of two optional cakes. There- fore, by analysing the code of the running program, we get that its space complexity is O (1)

Greedy algorithm, as the first hand, needs to traverse all cakes and store them in the list, and finally select the largest cake fast. So, the spatial complexity is O(n)

5.2 DP

DP, as a backhand, needs to constantly com- pare the size with the numbers of other cake blocks during each selection, and the results of each com- parison should be stored. There are n cakes in total. Each cake has to be compared with other N-1 cakes. Therefore, a two-dimensional array needs to be used to store data. Therefore, the space complexity is O(n2)

As the first hand, DP is much better than the second hand. When selecting first, it calculates each cake and stores it in a dictionary. When selecting for the first time, it always selects the one with the highest score, so the space complexity is

$$O(n^3)$$

5.3 DFS

When DFS is the first one, choose from all the cake pieces as the first one. Because the traversal of all cakes is in the form of binary tree, it is necessary to choose one from two pieces each time. So, the final spatial complexity is $\mathrm{O}(\mathrm{n}^2)$.

As a backhand, DFS only needs to choose from two pieces. After making the selection, it continues to look for the appropriate cake pieces. Because the decision tree we use is a binary tree, there are only two forks under a node. Therefore, the spatial complexity is $\mathrm{O}(2\mathrm{n})$. According to the definition of spatial complexity, the constant of spatial complexity can be omitted, so the final spatial complexity is $\mathrm{O}(\mathrm{n})$.

5.4 Minimax

Minimax algorithm is also based on binary decision tree.

When you start, you need to traverse all cakes horizontally, starting from each cake, and then traverse all other cakes downward. Each time you make a selection, you should choose one of the two blocks. So, the final spatial complexity is $O(n^2)$

When minimax is the backhand, select from the two pieces at the notch. After making the se- lection, continue to traverse all the cakes except this to find the appropriate cake pieces. The data structure used here is also a decision binary tree. Therefore, similar to DFS algorithm, the spatial complexity of minimax is $O(2n)$.According to the definition of spatial complexity, the constant of spatial complexity can be omitted, so the final spatial complexity is $O(n)$.

6. CONCLUSION

Through 500 game experiments, we make four algorithms compete with other algorithms and our own same algorithms. It is analysed and evaluated from the following aspects: hand in hand, victory rate, running time and storage space. In the end, we found that minimax had the highest winning rate regardless of time. But the fly in the ointment is that minimax ran for a long time and is not suitable for situations with time constraints.

If minimax is excluded, the greedy algorithm has a higher winning rate. Greedy algorithm not only has a high winning rate, but also performs well in running time and space.

DP algorithm is similar to greedy algorithm, but it is slightly inferior to greedy algorithm in winning rate and running time. At the same time, compared with the greedy algorithm, DP algorithm also occupies more storage space.

The worst is the DFS algorithm. This is a completely ambitious algorithm. At the same time, there is no special merit in running time and occupied space. Generally speaking, it is the most undesirable algorithm.

Our research on the algorithm in the cake cutting game can be applied to other commercial games, so as to improve the efficiency of commercial games and bring players a better experience.

With the continuous development of game artificial intelligence with implemented algorithms we tested, game is not the only place where this technology can play a role. It can be used to solve problems similar to game decision-making in the real world. In addition to the game industry, personal electronic assistants, recommendation systems, driverless driving, chip design, decision support and all other areas that need continuous decisions and decision-making are also the application scenarios of game AI technology.[5]

ACKNOWLEDGMENTS

Professor Bill Nace always gives us encouragement, determines the general direction of the overall paper for us, and enables us to complete the task on time. Teaching assistant Ziyi Wang always puts forward many new ideas for us, which makes our achievements more perfect.

The two authors are both the first author, and they have the same attribute to this paper.

REFERENCES

[1] Jose Luis Gonz´alez S´anchez, Natalia Padilla Zea, and Francisco L. Guti´errez.From usability to playability: Introduction to player centred video game development process. In Masaaki Kurosu, editor, Human Centred Design, pages 65–74, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] E. F. Provenzo. *Video Kids*. Video Kids.

[3] VJ Rayward-Smith. Introduction to the theory of complexity. *journal of the Operational Research Society*, 46(12):1507–1508, 1995.

[4] Jørgen Bang-Jensen, Gregory Gutin, and Anders Yeo. When the greedy algorithm fails. *Discrete optimization*, 1(2):121–127, 2004.

[5] Minqi Hu. Video games: an "alternative" training ground for artificial intelligence. *China Science Daily*, 2021-10-21(003).