

# Performance Comparison Breadth First Search and Depth Limited Search on Eight Box

<sup>1</sup>Ardhian Ekawijana

*Abstract---* Comparison of the performance of Breadth First Search and Depth Limited Search views of the four parameters. DLS seen from memory usage, iterations, and time is superior, because the solution is found to the left of the path. First Breath Search ahead in the optimal path to the discovery of the solution compared to DLS due to search the horizontal way.

*Key Word---* BFS, DLS, Eight Box, Performance

---

## I. INTRODUCTION

Artificial intelligence is a branch of informatics engineering, especially for cases that require skills or analysis that is usually owned by men. The development of this branch of science due to the growing ability of computers. Computation required in this branch of science requires a fast computer capability.

Artificial intelligence is already entered some aspects of the scholarly and society life today. The most striking aspect is the aspect of games. The rapid growth of the games in the community do not know the ages, from the young, elderly and young children there who have never played games. Medical Aspects growing in terms of introduction of disease, detection tools, automated infusion and others. Regulatory aspects of the industry both in the purchase of equipment, manufacture of arms, robot maids and the Internet of Things.

Artificial intelligence has some approximation method most commonly used method is searching for a search, determining the shortest path, and the solution of a game. Searching on the algorithm into 2 groups: Blind Search and Heuristic Search. Blind Search is a brute force method which is looking for every possible solution to generate all possible ways regardless of the information and just look at his pattern. Unlike the Heuristic Search, which uses information approximate solution in search of a solution [1].

Blind Search algorithm is simple in algorithm Searching SAT. The algorithms included are Blind Search Algorithm Breadth First Search (BFS), Depth First Search (DFS), Depth Limited Search (DLS), Iterative Deepening Depth-First Search (IDS), and Bi-Directional Search [1].

---

<sup>1</sup>ardhian.ekawijana@widyatama.ac.id,  
Widyatama University

This study aimed to compare the performance of BFS and DLS views of time, memory usage, iteration, and the optimal solution. The case studies are used in comparing the two methods are games Box 8. The programming language chosen is to use Java Programming.

## II. LITERATURE STUDY

### A. Blind Search

Blind Search is a search method in artificial intelligence that uses information only the starting point and end point as a reference. Search is brute force followed the rules every existing method. The mechanism of the algorithm is to raise a child node and then compare with the state goal [2].

Node in a blind search is divided into three types, namely [2]:

1. Fringe Node, an edge node that has been raised, unprocessed, yet compared to the goal and is usually called open node.
2. Visited Node, a node that has been raised, processed, compared to the goal and is usually called close nodes.
3. Undiscovered nodes, a node that has not been raised and unreachable from the root.

### B. Breath First Search

Breath First Search is a search method by finding the nodes horizontally. Node has been compared to the goal and it was not the solution raised its child nodes according to certain rules. Node compared with goal been horizontally, then down to the next depth in a tree.

This method ensures the results obtained are optimal result, but its weakness is that this method should save all the nodes that have been raised in the memory, to find a solution path if a solution is found on a node [1].

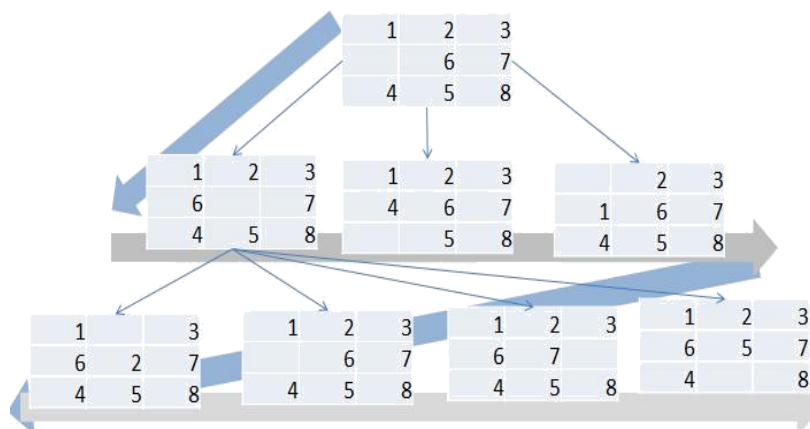


Figure 1: Breath First Search Algorithm Illustration

Figure 1 illustrates the flow of search Goal to discover a node on the tree horizontally. Every condition of the eight boxes counted as one node. Nodes that are not in accordance with Goal State will be raised child nodes. Search continues until there are the same node with Goal.

### C. Depth First Search

This algorithm is the opposite of the BFS was the focus of the search on the first horizontal. Depth First Search focus on vertical search to the bottom of the tree. The search starts from the left side and continued into. If the solution is in the left side of the tree then the algorithm will quickly find a solution, but if not then this algorithm will not find a solution [1].

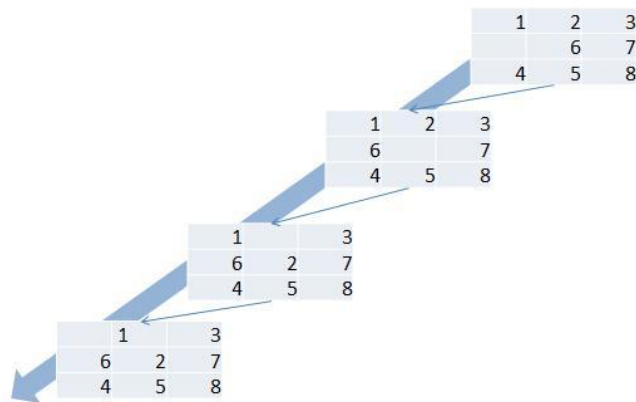


Figure 1: Depth First Search Algorithm Illustration

Figure 2 illustrates the workflow of an algorithm DFS. Search focus on the depth of the tree vertically downwards. Node will continue to be raised most left until you find a solution or to limit the depth of the governed. The solution to this algorithm is not guaranteed optimal results.

### D. Depth Limited Search

These algorithms address the weaknesses that are owned by DFS algorithm by limiting the depth that should be explored in the search. Determination of the depth of the deciding factor in this algorithm. Error in determining the depth can be no finding solutions, conversely if too large depth value will be longer in the search.

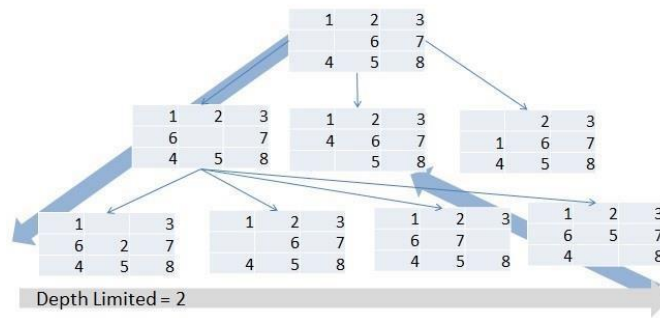


Figure 2: Depth Limited Search Algorithm Illustration

Figure 3 depicts the flow of a search using DLS. Searches were conducted vertically downwards until at a certain depth. Search direction will change when it reaches a predetermined depth [1].

*E. Eight Box*

Figure 3 depicts the flow of a search using DLS. Searches were conducted vertically downwards until at a certain depth. Search direction will change when it reaches a predetermined depth [1].

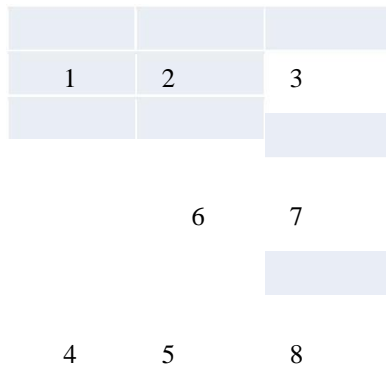


Figure 3: Random Eight Box

Figure 4 illustrates the condition of eight random from the box. The initial condition must be resolved by moving the numbers to the blanks.

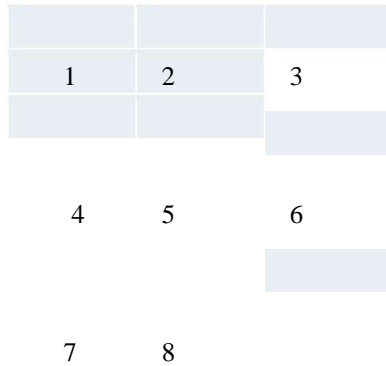


Figure 4: Final Condition Eight Box

Figure 5 illustrates the condition already ordered eight boxes with one box is empty. The final condition to be addressed by the search algorithm.

### III. DESIGN AND IMPLEMENTATION

#### A. Design

Software designed to support research using java programming, using the console program.

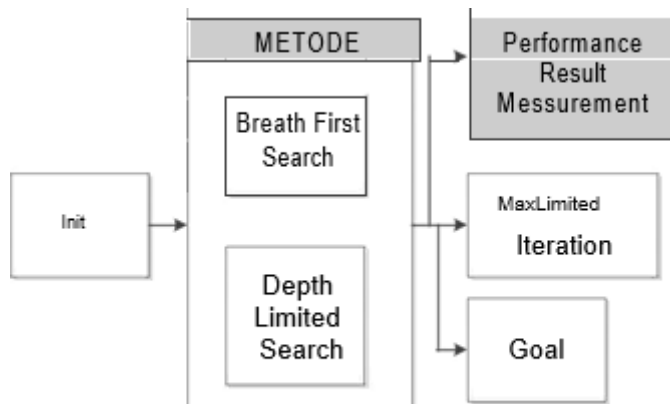


Figure 5 Research Design

Figure 6 describes the flow testing of two methods of searching. Originally conceived in the form of a box of eight random input. Input is processed by each of the existing method to find the solution or until the maximum iteration is determined. Bulge appeared also the measurement result.

### *B. Method Implementation*

BFS method implementations using java programing. The main procedures of BFS is as follows [3]:

```
void BreathFirstSearch(Kotak kotakInit, Kotak kotakTarget){ utilsRepository.addOpen(kotakInit);

    Kotak kotakfirst = new Kotak();

    int i=0;

    while (utilsRepository.getOpen()!=null && i<=100000){ i++;

        //System.out.println("iterasi ..... ke "+i);

        kotakfirst = (Kotak)utilsRepository.getOpen().elementAt(0);

        if (Arrays.deepEquals(kotakfirst.getSel(),kotakTarget.getSel())){ iterasi=i;

            tampilkanPath(kotakfirst);

            break;

            utilsRepository.removeOpen(0);

            utilsRepository.addClose(kotakfirst);

            utilsRepository.generateanakforkotak8Breath(kotakfirst);
implementation for DLS is as follows [3]:
void DepthFirstSearchLimitedDepth(Kotak kotakInit, Kotak kotakTarget){ utilsRepository.addOpen(kotakInit);

    Kotak kotaklast = new Kotak();

    int i=0;//, depth=0;
```

```
int maxDepth = 30;

while (utilsRepository.getOpen() != null && i <= 100000) { i++;

    kotaklast = (Kotak)utilsRepository.getOpen().lastElement(); if (kotaklast.getDepth() <= maxDepth){

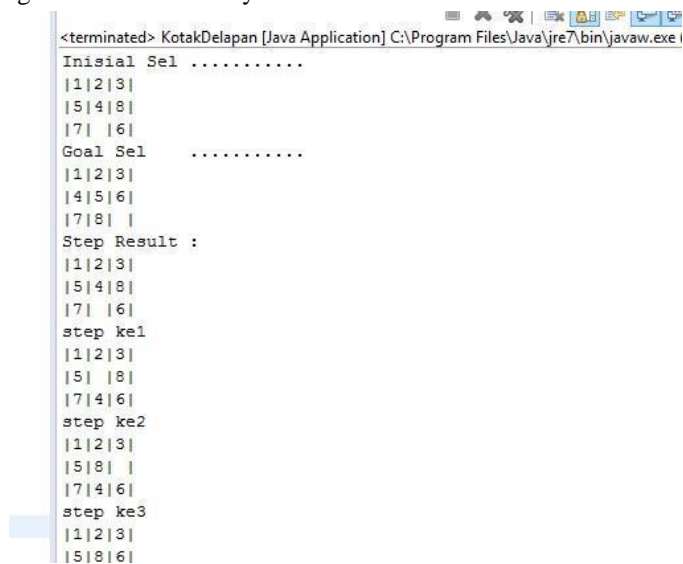
    if (Arrays.deepEquals(kotaklast.getSel(), kotakTarget.getSel())) { iterasi=i;
        tampilkanPath(kotaklast);
        break;
        utilsRepository.removeOpen(utilsRepository.getOpen().size()-1); utilsRepository.addClose(kotaklast);
        //depth++;
    if (kotaklast.getDepth() != maxDepth) utilsRepository.generateanakforkotak8Breath(kotaklast);
```

### C. Calculating Performance

Performance is calculated by looking at the time, Memory Usage, iterations and Optimal Path. The calculation of each performance viewed from the early beginning until the end of the iteration algorithm. The time taken from the start time and end time and then see the difference. Memory Usage views of how many nodes are generated during the iteration process takes place. Iteration seen from the number of repetitions to terkashir iteration. Optimal path seen from how many steps until the completion of the Init position Goal position [3].

### D. Human Interaction Computer

The interface used is using the console. Initially look like this:



```
<terminated> KotakDelapan [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe
Inisial Sel .....
11|2|3|
15|4|8|
17| 16|
Goal Sel .....
11|2|3|
14|5|6|
17|8| |
Step Result :
11|2|3|
15|4|8|
17| 16|
step ke1
11|2|3|
15| 18|
17|4|6|
step ke2
11|2|3|
15|8| |
17|4|6|
step ke3
11|2|3|
15|8|6|
```

Figure 7: Initial State

Figure 7 illustrates the initial condition of the box there are eight conditions expected end. Step Result is the solution results obtained after the program starts.

```
|7|8|6|  
step ke15  
|1|2|3|  
|4| |5|  
|7|8|6|  
step ke16  
|1|2|3|  
|4|5| |  
|7|8|6|  
step ke17  
|1|2|3|  
|4|5|6|  
|7|8| |  
step ke18  
Jumlah node yang tergenerate :24346  
Jumah step menuju goal :18  
Jumlah iterasi :16235  
execute time of BreathFirstSeach:54508
```

Figure 8: Final State

Figure 8 illustrates the final conditions of the application displays the data that exist. Applications featuring the performance of the algorithm itself.

#### IV. ANALYSIS

The algorithm is implemented with Java programming several experiments. Experiments done by selecting a different init state to produce the same goal state.

##### A. Memory Usage

Memory consumption becomes a good-sized failure of an algorithm. A good algorithm with at least one of which is the memory used by the algorithm.

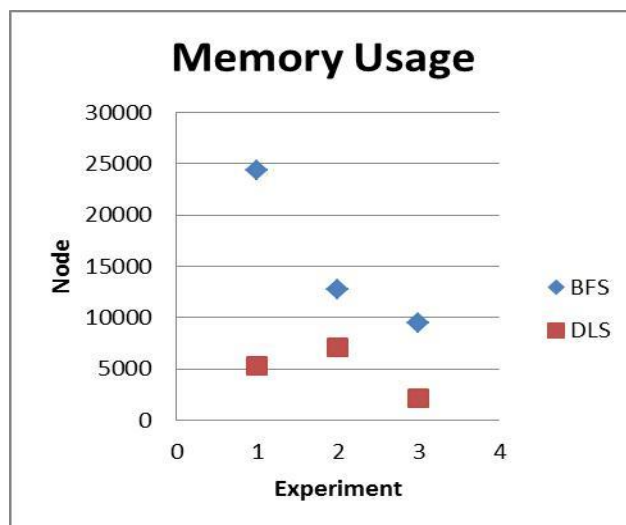


Figure 6: Memory Usage Experiment



In the chart above describes the memory consumption as seen from the number of nodes that are raised. DLS algorithm looks less wear memory. BFS algorithm more evoke the memory of a similar case.

### B. Solution Path

Solution Path algorithm is the ability to find the optimal path. A good algorithm is one that has the shortest path.

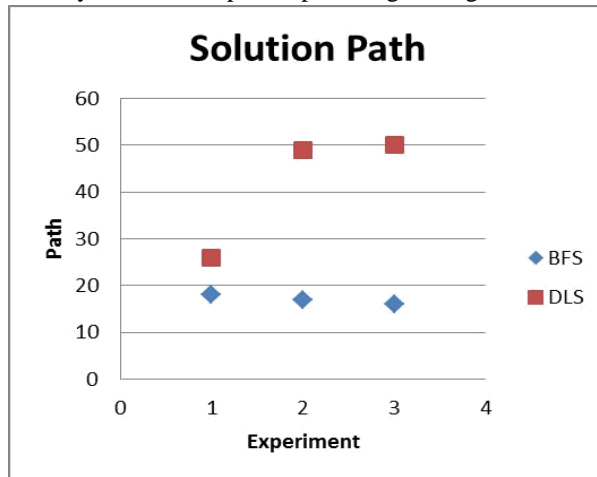


Figure 7: Solution Path

Solution Path in the three experiments above shows that to find a good solution looks BFS method. DFS algorithm has a long path solution than BFS.

### C. Iteration Summary

The number of iterations is one measure that could see how busy the computer is running.

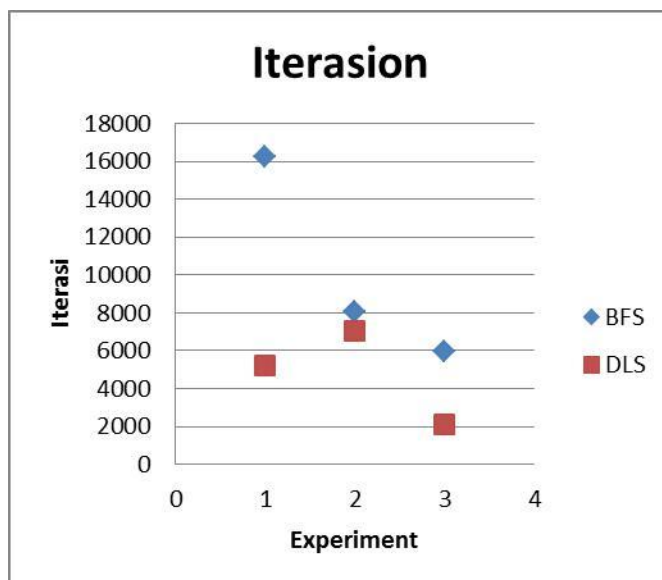


Figure 8: Iterasion

The number of iterations of several experiments that DLS better seen in the number of iterations. Iteration is a little portrait of the computer is not too busy.

### Time

Time becomes one parameter because better algorithmis the most quickly in finding Goal State.

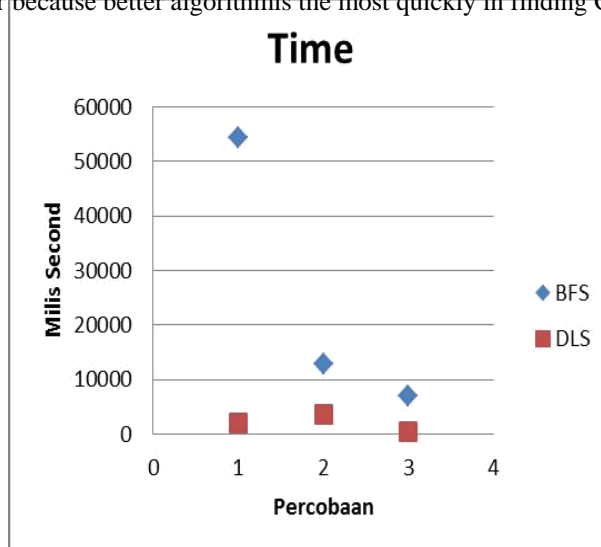


Figure 9: Execution Time

The time is needed to reach Goal State. DLS better than BFS because most solution obtained left most path.

## V. CONCLUSION

DLS looks better on memory consumption parameter, the number of iterations and time. This is evident because of the three experiments shown that the solution path is found on the left. BFS is discovered that an optimal solution path with the shortest path. DLS often do not find a solution path for vertical search pathways tend downwards, so it is not found the optimal solution path.

## REFERENCE

- [1] Suyanto, *Intelijensia Buatan*, Teknik Informatika, STT Telkom,2002.
- [2] Qu. R., *Introduction to Artificial Intelligence*, School of Computer Science, The University of Nottingham, 2001.
- [3] Riyanto. B, *Intelegensia Artifisial dan Komputasional*, STEI, ITB,2003