ECE1387 Assignment 3 Report

Brandon Nie
1003021482

1 Program Description

In this assignment the structure of my program is described below

1.1 Branching Structure

I have designed my decision tree as a binary tree. Each node in the tree will have the following information recorded:

- Level: depicts how many cells have already been assigned a decision up to the current node.
- Lower_bound: the lower bound of this node (partial solution).
- ID: the cell ID that this decision node represents.
- Left_num, Right_num: The number of cells inside the left or right partition in the current node (partial solution).
- Parent: pointer to the parent node of current node.
- Left: pointer to the left child.
- Right: pointer to the right child.
- Others: other data that helps with graphics and drawing the node.

1.2 Initial Solution Computation

For initial solution, I first sorted the cells based on fan out in increasing order. Then I partitioned the cells randomly, for 500 times. I would take the random partition with the lowest cut size to be my initial best solution.

1.3 Bounding Function

The Bounding function I use is:
Cut size of the current partial solution + the minimum cut size increase by adding each un-visited cell to either side of the current partitions

I have tried to use just the cut size of the current partial solution, and it did not provide a good result.

2 Equal Partition Size Results

The algorithm I implemented did not compute cct3 or cct4 in a reasonable amount of time (>15 minutes) and it would consume too much memory trying to compute cct3 and 4. Therefore, I only presented the results I got for cct1 and cct2. The graphics result of the tree traversal for cct1 and cct2 are shown in figure 1 and 2.

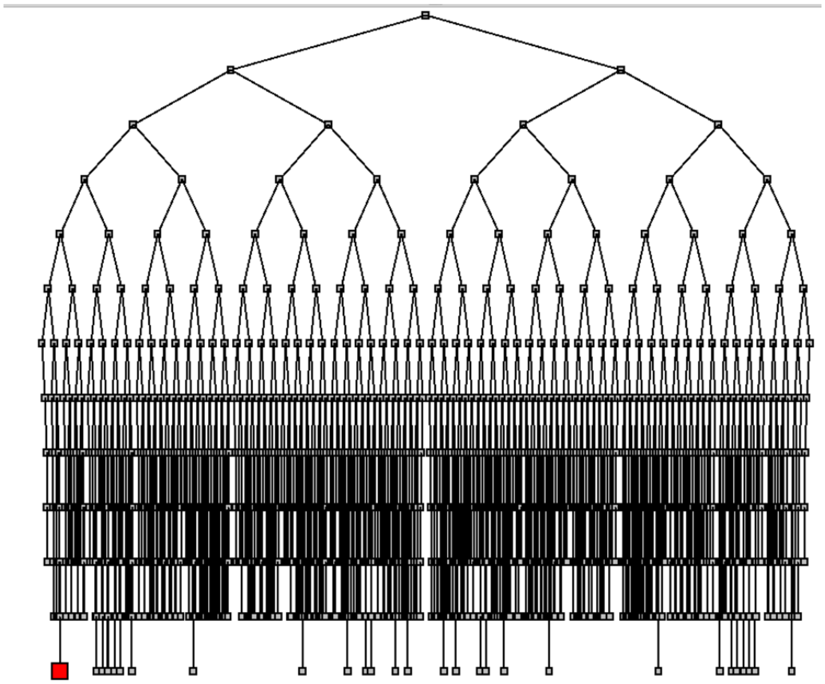

Figure 1. cct1 partition results.

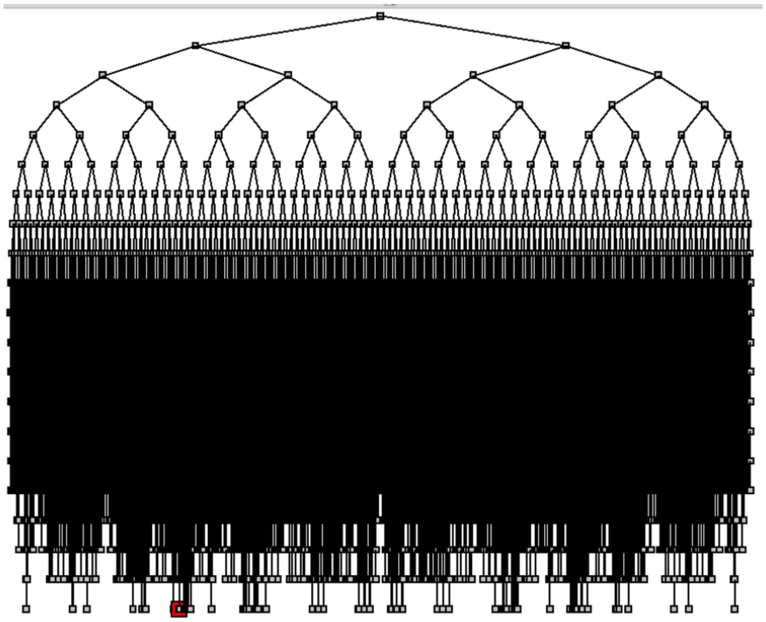

Figure 2. cct2 partition results.

The partitioning results including the number of nodes traversed, computation time, and cut size is included in table 1 and 2 for breadth-first search and lowest-bound-first search respectively.

	Cct1	Cct2	Cct3	Cct4
Run time	0.217s	5.98s	Not Computed	Not Computed
Nodes traversed	1835	84827	Not Computed	Not Computed
Cut size	28	42	Not Computed	Not Computed

Table 1. Breadth-first search results.

	Cct1	Cct2	Cct3	Cct4
Run time	0.22s	6.26s	Not Computed	Not Computed
Nodes traversed	1151	50225	Not Computed	Not Computed
Cut size	28	42	Not Computed	Not Computed

Table 2. Lowest-bound-first search results.

3 Unequal Partition Size Results

In this section, I have tuned my program to be able to allow either partition to be 60% of the total number of cells (equivalent to one side being 20% larger than the other). The graphics results for cct1 and cct2 are shown in figure 3 and 4. The computation data for breadth-first search and lowest-bound-first search is shown in table 3 and 4.

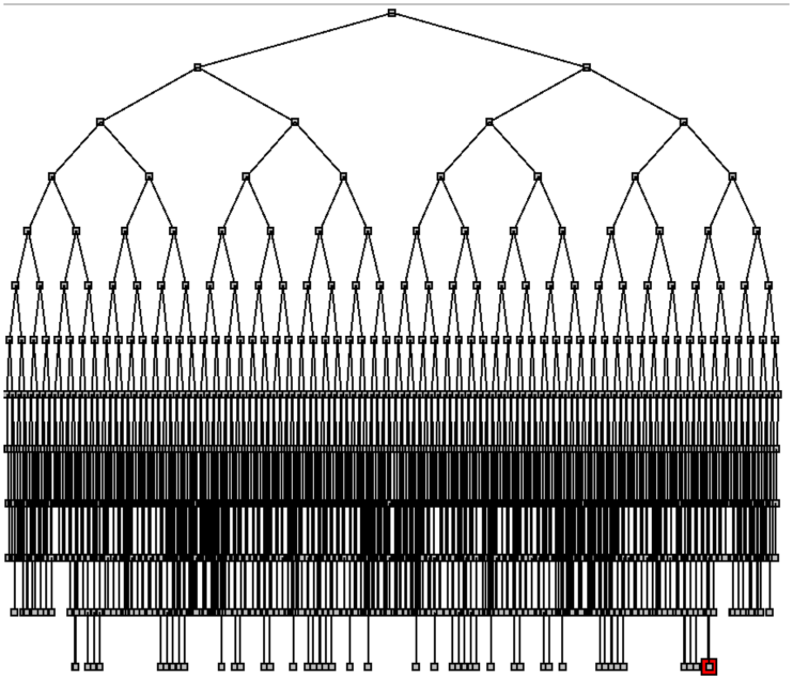

Figure 3. cct1 partition results.

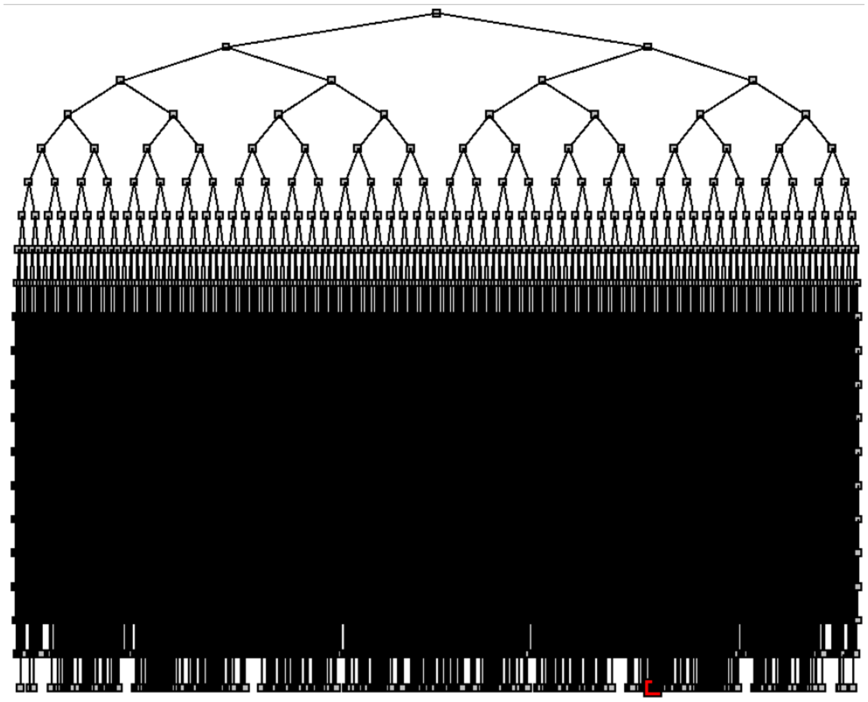

Figure 4. cct2 partition results.

	Cct1	Cct2	Cct3	Cct4
Run time	0.222s	7.65s	Not Computed	Not Computed
Nodes traversed	1941	124871	Not Computed	Not Computed
Cut size	25	37	Not Computed	Not Computed

Table 3. Breadth-first search results.

	Cct1	Cct2	Cct3	Cct4
Run time	0.223s	8.05s	Not Computed	Not Computed
Nodes traversed	1115	72975	Not Computed	Not Computed
Cut size	25	37	Not Computed	Not Computed

Table 4. Lowest-bound-first search results.

4 Discussions

From the above results one can conclude the following observations:

1. **Impact of Decision-Tree Traversal Order:**
   - Both Breadth-First Search (BFS) and Lowest-Bound-First Search (LBFS) yield the same final solution cut size.
   - LBFS, however, requires traversing significantly fewer nodes compared to BFS.
   - This suggests that the traversal order affects computational efficiency, with LBFS being more efficient in terms of the number of nodes explored.
2. **Effect of Unequal Partition Size:**
   - Introducing unequal partition sizes improved the final result cut size.
   - Unequal partition sizes expanded the solution space explored, leading to a better solution cut size.
   - However, this improvement comes at the cost of increased nodes traversed and longer run time.
   - There is a trade-off between solution quality (cut size improvement) and computational efficiency (nodes traversed and run time).