Code and Analysis 5 Figures with Legends

Jonathan Lee and Danielle Covarrubias

October 30, 2018

In these experiments, we measured the 3 following parameters against a range of input sizes for each problem:

1. Node Count
2. Frontier Count
3. Depth

By capturing the number of nodes expanded, we are able to measure the time complexity of each algorithm. In order to measure space complexity of each algorithm, we also kept track of the max frontier count in each experiment. Knowing the depth that the program reached until it ran out of time or memory is a good indicator of how far each algorithm can get based on the size of the input.

The parameters used for the simulated annealing experiments were as follows:

Parameter 1 (default)
max_time = 100 (arbitrary maximum time for any given task)
n = 20 (number of jobs)
p = 5 (number of people)

Parameter 2 (twice as many people as default)
max_time = 100 (arbitrary maximum time for any given task)
n = 20 (number of jobs)
p = 10 (number of people)

Parameter 3 (half as many jobs as default)
max_time = 100 (arbitrary maximum time for any given task)
n = 10 (number of jobs)
p = 5 (number of people)

The largest possible input size for both Queens and Towers was experimentally found to be:

N-Queens
IDDFS Tree – 11 Queens (3.5 hours to compile)
BFS Tree – 8 Queens (stopped running after 1.5 hours)

Towers of Hanoi
BFS Graph - 11 disks (timed out after 1 hour)
DFS Graph - 12 disks (timed out after 1 hour)
IDDFS Graph - 5 disks (timed out after 1 hour)

Table 1: NQueens - BFS Tree Search vs. IDDFS Tree Search

	Max Depth	**Node Count**	**Max frontier Count**
BFS Tree Problem Size			
4	4	201	149
5	5	1,141	911
6	6	22,093	18,409
7	7	182,610	156,521
8	6	1,271,436	1,155,848
IDDFS Tree Problem Size			
4	4	97	7
5	5	363	11
6	6	2,475	16
7	7	14,225	22
8	8	117,344	29
11	11	109,761,332	56

As shown in Table 5, IDDFS tree search outperforms BFS tree search on every single problem size in terms of nodes generated and the maximum number of nodes in the frontier, highlighting its superior runtime and memory use over the BFS tree algorithm.

Table 2: Simulated Annealing in Job Scheduling - Default: time = 100, jobs = 20, people = 5

	Total iterations	**Moves to better states**	**Moves to worse states**	**No moves**
Default				
Trial 1	364,001	199,018	55,034	109,949
Trial 2	364,001	205,758	54,470	103,773
Trial 3	364,001	208,336	55,816	99,849
Trial 4	364,001	203,809	55,338	104,854
Trial 5	364,001	207,608	55,129	101,264
Average	364,001	204,906	55,157	103,938
people = 10				
Trial 1	364,001	220,800	43,134	100,067
Trial 2	364,001	222,877	43,830	97,294
Trial 3	364,001	219,769	43,966	100,266
Trial 4	364,001	226,104	42,036	95,861
Trial 5	364,001	227,113	42,555	94,333
Average	364,001	223,333	43,104	97,564
jobs = 10				
Trial 1	364,001	205,941	60,374	97,686
Trial 2	364,001	199,641	62,261	102,099
Trial 3	364,001	201,226	60,857	101,918
Trial 4	364,001	200,629	61,497	101,875
Trial 5	364,001	202,767	60,972	100,262
Average	364,001	202,041	61,192	100,768

From the results of Table 2, doubling the number of people in job scheduling using simulated annealing, on average, results in a more efficient state whereas halving the number of jobs tends to yield a worse state when comparing them to the default parameters.

Table 3: Towers BFS-Graph

# of disks	**3**	**6**	**8**	**10**	**11**
Total Node Count	54	1932	18660	173052	514704
Max Frontier Count	6	42	170	682	1024
Depth	7	63	255	1023	2044

When BFS-graph search was used to find a solution for Towers, it can be seen in Table 3 that the program times out after 11 disks.

Table 4: Towers DFS-Graph

# of disks	**3**	**5**	**9**	**11**	**12**
Total node count	39	363	29523	265719	327855
Max frontier count	8	63	4925	44292	54647
Depth	13	121	9841	88573	109285

In Table 4, DFS-graph search times out after 12 disks, which is a better time complexity improvement from BFS.

Table 5: Towers IDDFS-Graph

# of disks	**3**	**4**	**5**
Total node count	1495	12801757	350694110
Max frontier count	14	30	36
Depth	7	15	18

Solving Towers using ID-DFS graph search can be seen in Table 5, where the program timed out after 5 disks; the time complexity is significantly bigger compared to BFS and DFS.

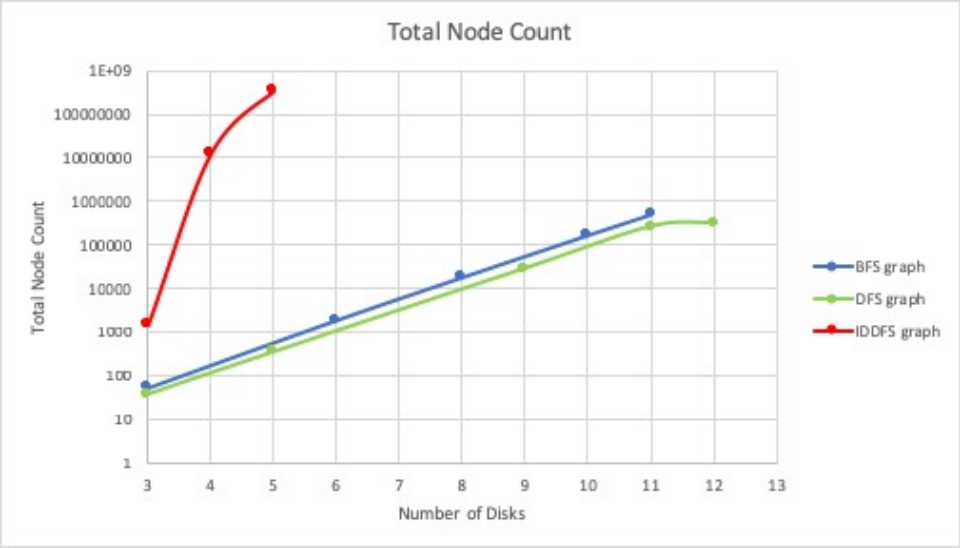

Figure 1: Towers Experiments Total Node Count Results

Looking at Figure 1, IDDFS-graph exponentially expands more nodes compared to the linear trends of BFS-graph and DFS-graph, resulting in IDDFS-graph to have a much bigger time complexity than the other two algorithms.

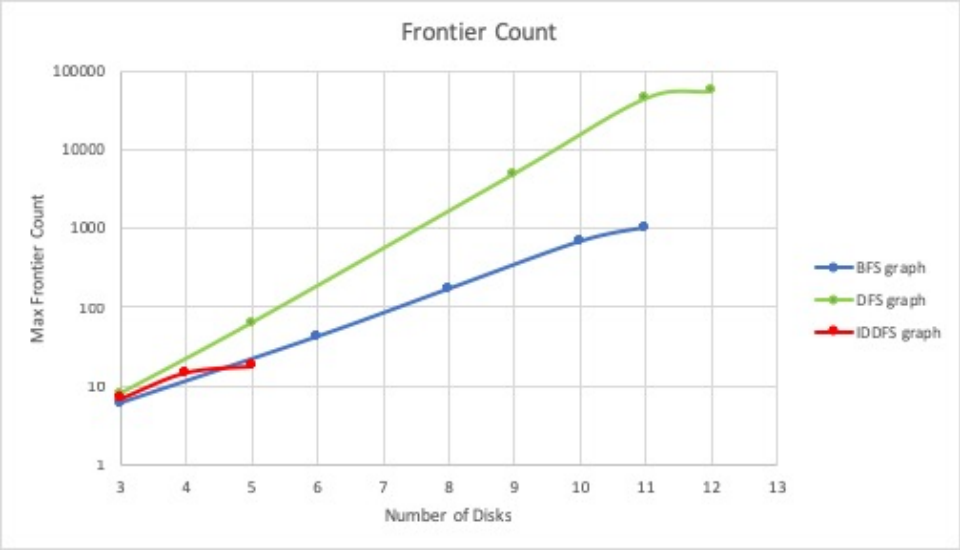

Figure 2: Towers Experiments Max Frontier Count Results

According to Figure 2, DFS-graph had the largest frontier count, which means it would essentially have a bigger space complexity than the other two; all 3 algorithms timed out before they ran out of memory.

Figure 3: Towers Experiments Depth Results

Based on Figure 3, DFS-graph search always reached the deepest nodes in the graph, which is where the solution can be found the quickest, hence why it performed the best. IDDFS timed out quickly because it would never reach the solutions at the depths of the graph.

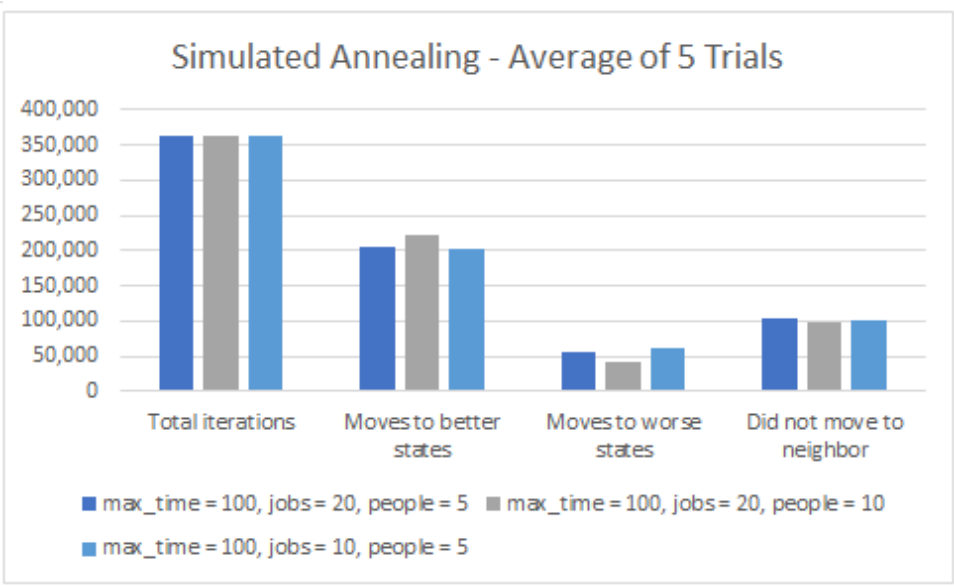

Figure 4: Simulated Annealing Experiments Results

Figure 4 displays an improvement in returning a more optimal state when doubling the number of people and comparing the results to the default parameter with regards to job scheduling using simulated annealing.

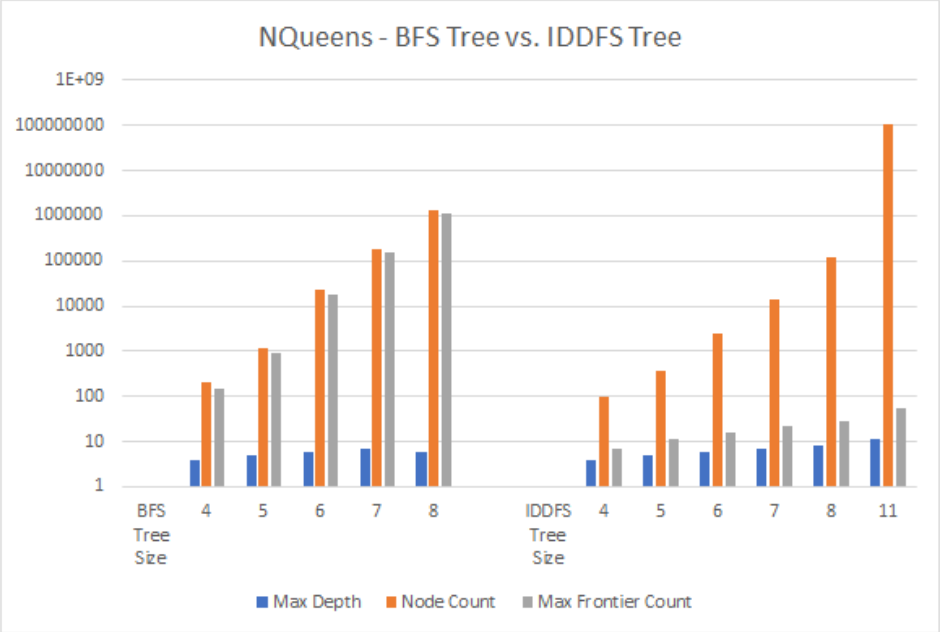

Figure 5: NQueens Experiments Results

Figure 5 highlights the great difference between space and time complexity of IDDFS tree search and BFS tree search, respectively, as applied to the problem of NQueens.