COMP20003 - Algorithms and Data Structures

Assignment 3- Experimentation

Introduction

The experimentation is going to discuss the performance of Al algorithm on playing Peg Solitaire. The algorithm is based on Depth First Search (DFS) strategy which is using a stack to decide explored nodes and a hash table to avoid duplicate cases, eventually it is expected to find out the best solution with limited budgets. The following table contains the important data including the number of pegs left, generated nodes, expanded nodes, expanded/second, total execution time and ratio of generated/expanded nodes for each layout and each max budget of 10K, 100K, 1M, 1.5M.

Layout 0 (3 pegs)

Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	1	2	2	44	0.045158	1.000000
100,000	1	2	2	47	0.041672	1.000000
1,000,000	1	2	2	49	0.040318	1.000000
1,500,000	1	2	2	46	0.043138	1.000000

Layout 1 (4 pegs)

Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	1	3	3	74	0.040454	1.000000
100,000	1	3	3	71	0.042198	1.000000
1,000,000	1	3	3	71	0.041888	1.000000
1,500,000	1	3	3	66	0.045426	1.000000

Layout 2 (7 pegs)

Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	1	8	7	166	0.041963	1.142857
100,000	1	8	7	157	0.044426	1.142857
1,000,000	1	8	7	174	0.040214	1.142857
1,500,000	1	8	7	162	0.042999	1.142857

Layout 3 (17 pegs)

Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	1	10,282	3,541	54,705	0.064728	2.903700
100,000	1	10,282	3,541	50,311	0.070382	2.903700
1,000,000	1	10,282	3,541	54,195	0.065338	2.903700
1,500,000	1	10,282	3,541	50,461	0.070173	2.903700

Layout 4 (32 pegs)

Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	1	2,418	1,065	21,500	0.049534	2.270423
100,000	1	2,418	1,065	21,012	0.050683	2.270423
1,000,000	1	2,418	1,065	21,852	0.038736	2.270423
1,500,000	1	2,418	1,065	21,925	0.048574	2.270423

Layout 5 (36 pegs)

Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	4	26,495	10,000	92,208	0.10845	2.649500
100,000	3	359,818	100,000	213,799	0.467729	3.598180
1,000,000	2	4,488,464	1,000,000	259,411	3.854433	4.488464
1,500,000	1	4,898,609	1,090,275	277,066	3.93507	4.493003

Layout 6 (44 pegs)

Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	5	29,368	10,000	98,229	0.101802	2.936800
100,000	4	374,378	100,000	204,160	0.489811	3.743780
1,000,000	3	4,481,233	1,000,000	274,582	3.641888	4.481233
1,500,000	3	7,020,668	1,500,000	274,564	5.463202	4.680445

Layout 7 (38 pegs)

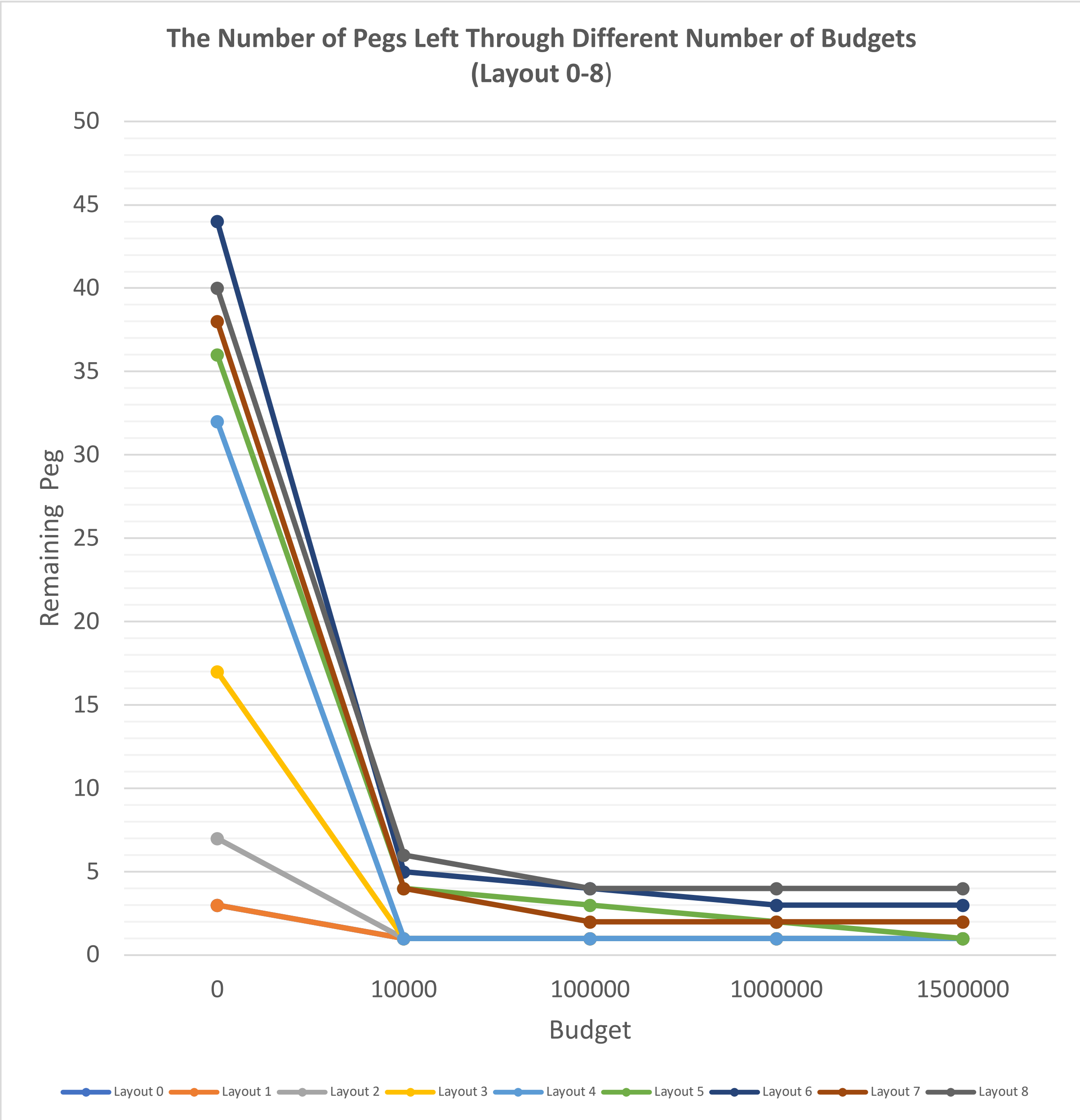
Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	4	32,469	10,000	85,480	0.116986	3.246900
100,000	2	386,440	100,000	200,250	0.499375	3.864400
1,000,000	2	4,790,308	1,000,000	257,469	3.883954	4.790308
1,500,000	2	7,173,504	1,500,000	241,636	6.207679	4.782336

Layout 8 (40 pegs)

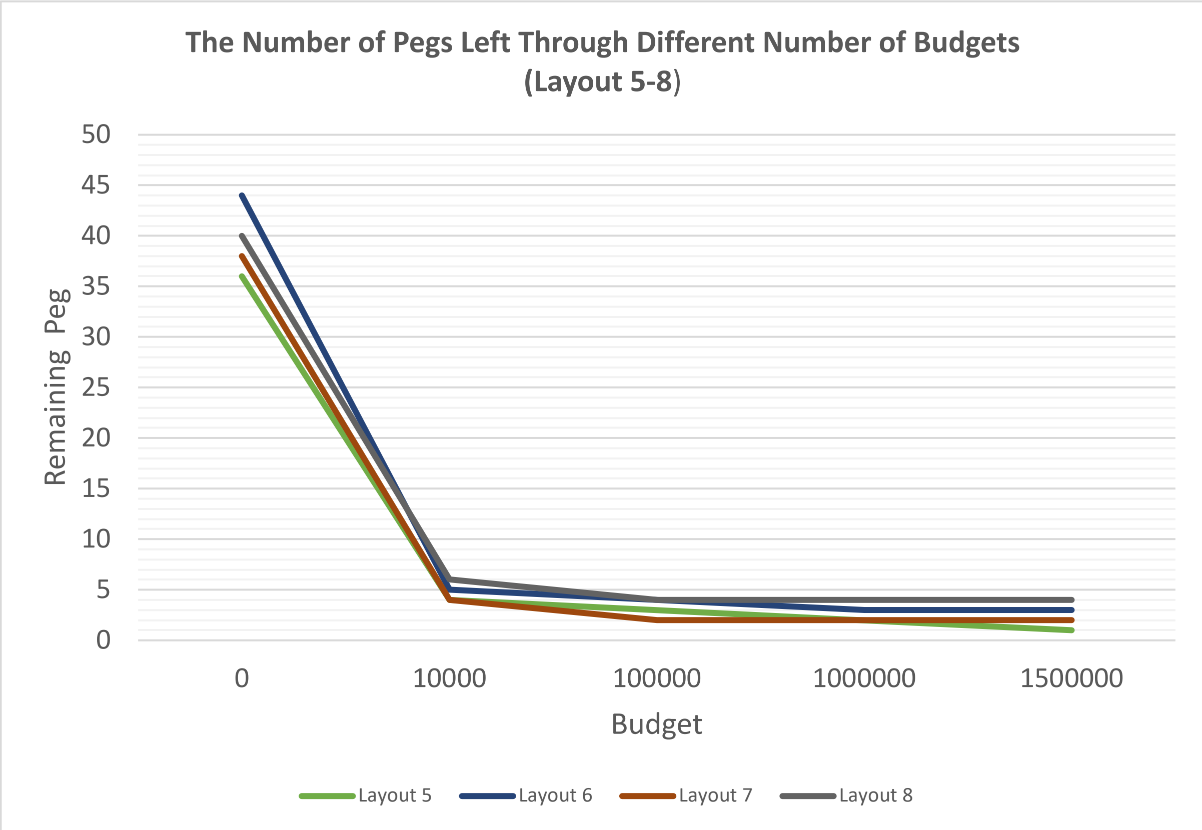
Budgets \ Results	Number of Pegs left	Generated Nodes	Expanded Nodes	Expanded/Second	Total Execution Time (seconds)	*Ratio of Generated/Expanded
10,000	6	27,562	10,000	97,988	0.102053	2.756200
100,000	4	349,921	100,000	212,286	0.471062	3.499210
1,000,000	4	4,073,028	1,000,000	286,336	3.492394	4.073028
1,500,000	4	6,361,454	1,500,000	219,024	6.848549	4.240969

(Table 1)

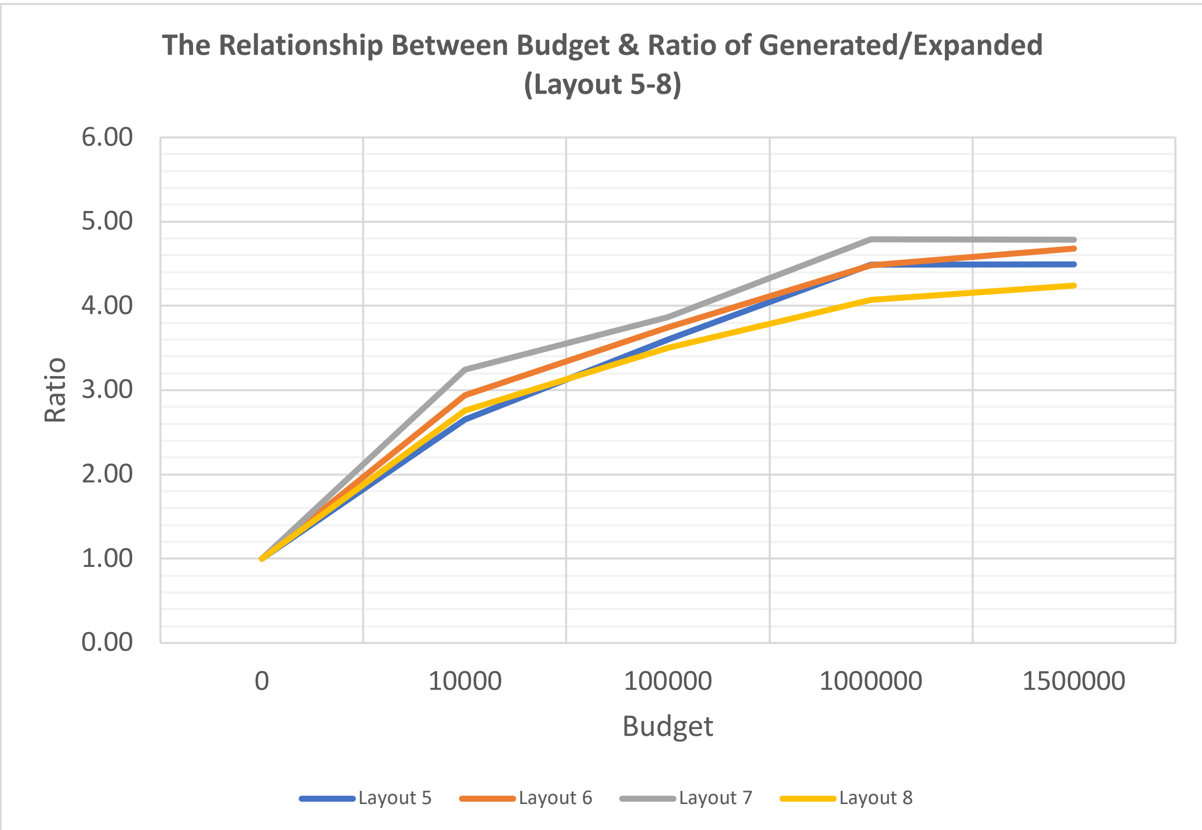
To be clearer, the following plots show the number of pegs left through different number of budgets, and the relationship between budget and ratio of generated nodes to expanded nodes.



(Figure 1)



(Figure 2)



(Figure 3)

Analysis

The experimental results reveal that the question which has less than thirty initial pegs can be easily solve by under 10K expanded nodes. However, although the difference of initial pegs between layout 4 and layout 5 is four pegs, the layout 5 required nearly 1,090,000 more budgets than layout 4 to find the solution. Also, layout 8 initially has less pegs than layout 6, and yet layout 6 left less pegs at the end of 1.5M budgets. Therefore, besides the difference of number, the shape of board may also influence the performance on the algorithm. From Figure 1, we can see that most of the layouts can be figured out the solution which has under five remaining pegs by using 10K budgets. It indicates that the algorithm can easily eliminate the number of pegs quickly. However, to get a better solution is harder and harder through the solving process. It can be seen more clearly in Figure 2 which only plots the last four layouts. The solution quality is lower as the budgets increasing. It means that to find out the best solution, the cost of budgets is very expensive and maybe not efficient. The reason might be that DFS strategy need to traverse back and discover other paths. Without any optimization, the hash table would keep recording the similar rotationally board which is no need to be test further since they are in the same shape, and it would increase the explored nodes which is going to increase the demand of budget as well. To estimate the duplicate probability, the ratio of generated nodes to expanded nodes is increasing through the growth of budgets (see Figure 3). Since a proper (unseen) nodes would require more generated nodes, the trend highlights that more duplicate states would be created as the DFS is attempting more possibilities. On the other hand, if an optimization can be made for reducing duplicate cases (including rotational and symmetrical), the efficiency of finding the best solution might be higher.

Conclusion

Overall, the AI algorithm using Depth First Search strategy can quickly reduce the remaining pegs in 10K budgets. Yet, to find out the best solution (one peg left condition), it requires far more budgets than expected. Thus, there might be a dilemma of giving up certain accuracy and saving some budgets eventually. Additionally, the frequency of duplicate board is increasing as the increment of budgets. To optimize the programming, it can be upgraded by classifying rotational and symmetrical boards as a seen state. In this way, it would push only the necessary generated nodes into the stack without duplicate boards and it would make the programming to use the limited budgets more efficiently.