tuankhoin

Tuan Khoi Nguyen-1025294

Assignment 2: Experimentation

Introduction:

For Assignment 2, we have to set up an AI agent based on a variant of Dijkstra's algorithms and use it to play the Pac Man game and get the highest score possible for different given budgets.

Theory and method:

Dijkstra's Greedy Algorithm is based on the idea that any sub-path of the shortest path is also the shortest path. By implementing this idea to find the most rewarding direction, the movement of Pac Man is determined by the AI. This is being implemented by getting the reward of each movement, and either add them up or get the average depends on the propagation. Then, the first movement node with the most rewarding value, will be chosen to be the next move. This action will repeat for every step made.

For each step, the Reward value is calculated with this formula:

$$r(n) = (h(n) + score(n) - score(nParent)) \times \gamma^d$$

With $h(n) = isBecomingInvincible \times 10 - isLosingALife \times 10 - isGameOver \times 100$

And $\gamma = 0.99, d = depth$

Then, the total reward of the initial movement node is calculated based on the propagation type:

MAX reward: will add up the reward r(n) of each corresponding step, propagating all the way back to the initial move to get the total reward.

AVG reward: will get the overall average of r(n) of all first move node's children.

And after running out of budgets, the movement is then chosen by picking the one with the highest rewarding value out of 4.

Experiment:

To test the AI, the experiment is carried out for 3 levels 1,2 and 3, for both propagation types with 4 different given budgets of 10, 100, 1000, 2000. Each type of experiment will take 3 attempts. In total, 72 plays will be carried out.

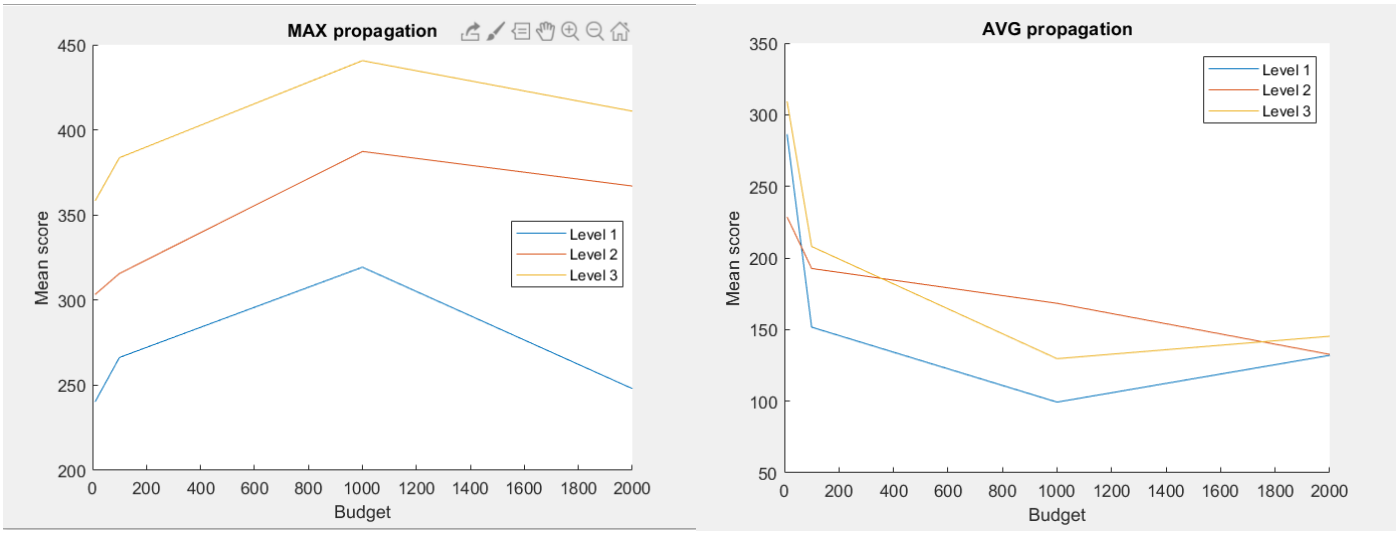
Results:

Over the page is the experiment results table and graphs:

Level	Propagation	Budget	Mean	Max Depth	Total Expanded Nodes	Total generated nodes	Time	Expanded nodes/ second	Score
1	MAX	10	240.3333	17	6091	13696	59.62969	102.1471	272
				23	9656	21072	94.87159	101.7797	251
				18	4284	9598	42.21737	101.4748	198
		100	266.3333	26	92354	200863	91.76176	1006.454	273
				39	87588	192122	87.53747	1000.577	291
				39	133013	287780	132.5902	1003.189	235
		1000	319.3333	53	840070	1858701	84.12718	9985.714	276
				39	705069	1562520	71.0723	9920.447	313
				36	737077	1626563	73.62807	10010.82	369
		2000	248	49	1322049	2864590	73.31582	18032.25	254
				38	1332032	2908268	72.53484	18364.03	318
				35	802043	1738510	43.87473	18280.3	172
	AVG	10	286.3333	17	6649	14759	65.85955	100.9573	249
				20	8075	17766	80.28733	100.5763	334
				14	5844	12757	57.84656	101.0259	276
		100	151.6667	25	75030	162914	74.58307	1005.992	282
				23	77702	172174	76.80304	1011.705	91
				23	66000	143826	65.27533	1011.102	82
		1000	99.33333	32	883005	1969502	87.37609	10105.8	123
				30	537009	1207117	53.15602	10102.51	88
				39	408017	912410	40.91356	9972.659	87
		2000	132	39	1616057	3601398	90.65342	17826.76	163
				26	1788119	4018630	102.6779	17414.84	77
				37	1680085	3749461	94.24214	17827.32	156
2	MAX	10	303.3333	18	5461	11795	54.81	99.6351	427
				14	2676	6011	26.66521	100.3555	204
				18	6293	13820	62.71656	100.3403	279
		100	315.6667	29	67600	151310	66.68734	1013.686	340
				28	55300	124496	54.59949	1012.83	272
				44	48914	110355	48.4607	1009.354	335
		1000	387.3333	45	636096	1442062	64.45622	9868.653	443
				32	521006	1171262	51.77868	10062.17	359
				46	643009	1442906	63.85819	10069.33	360
		2000	367	32	1068007	2386539	58.72229	18187.42	379
				46	1586028	3535226	89.85682	17650.61	479
				40	1194086	2692269	67.17266	17776.37	243
	AVG	10	228.6667	17	7250	15918	71.75732	101.035	292
				17	3836	8467	38.25245	100.2812	137
				16	3773	8386	37.65593	100.1967	257
		100	192.6667	43	35823	78180	36.6499	977.438	138
				34	59579	131044	59.63036	999.1386	234
				40	49224	110216	48.93999	1005.803	206
		1000	168.3333	32	537000	1224890	53.06675	10119.33	122
				24	404018	921319	40.28036	10030.15	191
				35	912002	2025043	90.32365	10097.05	192
		2000	132.6667	30	1178011	2666773	67.52811	17444.75	174
				19	288010	645337	17.33714	16612.32	29
				28	882042	2008694	50.05641	17620.96	195

3	MAX	10	358.3333	19	8948	20085	88.76383	100.8068	366
				19	4407	9679	43.59619	101.0868	233
				25	10796	24105	106.9437	100.9503	476
		100	383.6667	32	96606	213677	95.6909	1009.563	389
				25	92407	203716	91.47225	1010.219	334
				36	105371	230536	104.6991	1006.417	428
		1000	440.6667	50	2148084	4651774	213.8266	10045.92	689
				26	772000	1728091	76.33991	10112.67	365
				42	820000	1822322	81.03987	10118.48	268
		2000	411	47	2128049	4681244	117.1925	18158.57	398
				42	2244121	5002090	130.5014	17196.15	422
				41	2138031	4706473	119.7137	17859.54	413
	AVG	10	309.3333	14	7496	16714	74.39374	100.7612	349
				15	6440	14359	63.65207	101.175	397
				12	5888	13150	58.35884	100.893	182
		100	208	21	72834	163193	72.77771	1000.773	180
				17	37301	82358	36.93907	1009.798	69
				41	110072	244995	109.6728	1003.64	375
		1000	129.6667	30	552001	1214537	54.67265	10096.47	150
				24	503000	1086767	49.57423	10146.4	144
				22	478003	1046635	47.5073	10061.68	95
		2000	145.3333	20	1048013	2301352	57.79053	18134.68	115
				25	1590039	3521043	87.57722	18155.85	172
				19	1098000	2464439	61.12215	17964.03	149

Performance Graph for 2 propagation types:



Discussion:

- The columns in the table reflects the time complexity of Dijkstra's algorithm, which is $O((V + E) \times log(V))$.

- For low budget (of around 10), the average propagation preforms slightly better than the maximum propagation. When the budget increases, maximum propagation gives better score than the average one, as the latter improves performance to its peak at 1000 then slowly decreases, while the former decreases its effectivity to 1000, then slightly improves again. It can be concluded that for the budget range of 10 to 2000, the maximum propagation performs better overall and hence should be chosen for this range of budget.

- A reasonable explanation for this is because the average propagation also takes the death moves into account of average, which in total will not make a big change if other children nodes are safe moves. The maximum propagation, on the other hand, will stop the adding of scores if a move leads to a life lost, making its initial move less trustworthy.

- One problem that affects the accuracy of the AI is that it is only able to calculate a few consecutive safe moves (reflected from the majority of max depths of each move), making it unable to reach far food when approaching dead end and oscillates there. Another mentionable factor that reduces the accuracy of the AI agent for both propagations is the movement of the ghosts, which are generated randomly. In conclusion, the AI's accuracy is satisfied, but can be further improved.

Other improvement trials and modifications:

To make the AI move more efficient and avoid oscillation, I have added a logic during choice-making to make the AI move better when making a tie break: prioritize choice of move in the decreasing order: perpendicular to recent move, the same direction as recent move. To do this, I have modified the get_next_move function to have another input: move_t recent_move, the previous move of PacMan, which will be set randomly if it is the first move of the level.

Results: The AI improvement works well on personal device initially. However, when brought up to JupyterHub, the AI did not run well and will stuck if bumps a wall in vertical direction. Even though not being found yet, the problem is being predicted to be either a logical error, or a terminal difference in JupyterHub.