Automated Decision Making: Final Project

Luca Lumetti
244577@studenti.unimore.it

July 2, 2021

1 Introduction

In this project, I've tried to face the Max-Mean Dispersion Problem using Tabu Search guided by a deep reinforcement learning algorithm during the dispersion phase.
Given a complete graph G(V, E) where each edge has associated a distance or affinity d_{ij}, the Max Mean Dispersion Problem is the search of a subset of vertex $M \subset V, |M| >= 2$ which maximize the mean dispersion, calculated as follow:

$$MeanDispersion(M) = \frac{\sum_{i<j;i,j \in M} d_{ij}}{|M|}$$

This problem is known to be strongly NP-hard [**?**] and has the characteristic that the subset M does not have a fixed size, but can vary between 2 and $|V|$.

2 RLTS

In 2020, Nijimbere et al. proposed an approach based on the combination of reinforcement learning and tabu search, named RLTS. The main idea is to use Q-Learning to build a high-quality initial solution, then improving it with a one-flip tabu search.
Then Q-Learning algorithm is trained during the search of the solution, by giving a positive or negative reward if the node remain or not in the solution after the tabu search.

3 DQNTS

My proposed solution is to use a Deep Q-Network to generate the initial solution and tabu search to improve it, hence the name DQNTS. The approach is similar to [2], the network learns a heuristic to build the initial solution, then uses a one-flip tabu search to improve that solution. The difference is that the DQN is pre-trained, hence there is no need to start with a random solution and I can generate a high-quality initial solution from the beginning.

3.1 Network Architecture

The network architecture is based on [1], the implementation and the hyperparameters settings can be seen on github.com. The state2tens embedding is done with 4 features extracted from each node, which are:

- 1 if the node in the solution, 0 otherwise
- the sum of all edges connected to the node
- the sum of all edges connected to the node and the solution nodes
- the sum of all edges connected to the node and the nodes not in the solution

3.2 Network training

To train the network, first, we construct a feasible solution using an ϵ-greedy strategy, stopping when no positive rewards are predicted by the network. Then the solution is given to the one-flip tabu search which improves it and returns the best solution. For every node in the initial solution which remains in the final solution after tabu search, a reward of $+1$ is given, otherwise, the reward is -1. The network has been trained over 10 different instances (MDPIx_35 and MDPIIx_35, $1 <= x <= 10$) for 5001 episodes.

3.3 Tabu Search

The tabu search implementation is the same as in [2], with the only difference in the parameter $\alpha = 100$ instead of $\alpha = 50000$. My implementation couldn't finish even a single iteration in the time limit imposed with the parameter α proposed in the paper. This made me think that my implementation is way slower, but still, I left the same time constraints as the results were good enough.

3.4 General Algorithm

DQN is used during the construction of the initial solution: for all the nodes, the network estimates the reward, then all the values estimated get interpolated in the range $[-1, +1]$. All the nodes with a value $>= 0$ are named as "good nodes". Among these "good nodes", a random amount is taken to construct the initial solution. Picking one node at a time would result in a better solution, but this approach was too slow for large graphs. This solution is then processed with a one-flip tabu search until no best solutions are found for $\alpha = 100$ iterations in a row. Now a new initial solution is generated and the process is repeated until the time limit is not violated. Finally, the best solution found is returned.

4 Results

The algorithm has been tested over the same 60 instances of [2], downloaded from here and I've compared the results against the ones obtained by RLTS. The time limit has been set to 10s for $N <= 150$, 100s for $500 <= N <= 1000$, 1000s for $N = 3000$ and 2000s for $N = 5000$.
The RLTS algorithm seems to perform better on smaller graphs while my algorithm seems to perform better on larger graphs. The word "seems" is used here because the random nature of my algorithm can lead to different results when executed multiple times. To have more reliable results, more tests over the same instances should be made. This has been done in [2], but that would require me a computational time that I don't have.
The CPU used by the RLTS algorithm is an Intel Xeon Processor E5-2670 @ 2.5GHz, 20 core, and the GPU is Nvidia Tesla K20m I tested my algorithm on Intel Core i7-3610QM @ 2.30GHz, 4 core, and no GPU. The better results on larger graphs probably come from a better first initial solution than RLTS, which is fundamental on graphs of this size as the tabu search part take nearly all the time limit.

Instance	N	RLTS	DQNTS	Obj. Gap
MDPI1_150	150	45.920	45.920	0.000
MDPI2_150	150	**43.392**	43.386	0.006
MDPI3_150	150	**40.046**	40.037	0.011
MDPI4_150	150	44.044	44.044	0.000
MDPI5_150	150	42.479	42.479	0.000
MDPI6_150	150	43.723	43.723	0.000
MDPI7_150	150	46.077	46.077	0.000
MDPI8_150	150	42.451	42.451	0.000
MDPI9_150	150	42.480	42.480	0.000
MDPI10_150	150	41.798	41.798	0.000
Instance	**N**	**RLTS**	**DQNTS**	**Obj. Gap**
MDPI1_500	500	**81.277**	81.134	0.143
MDPI2_500	500	78.610	78.610	0.000
MDPI3_500	500	**76.301**	76.087	0.214
MDPI4_500	500	**82.332**	82.229	0.103
MDPI5_500	500	**80.354**	80.195	0.150
MDPI6_500	500	81.249	81.249	0.000
MDPI7_500	500	**78.165**	77.323	0.842
MDPI8_500	500	**79.140**	78.931	0.209
MDPI9_500	500	**77.421**	77.352	0.069
MDPI10_500	500	81.310	81.183	0.000

Instance	N	RLTS	DQNTS	Obj. Gap
MDPI1_750	750	**96.651**	94.949	1.702
MDPI2_750	750	**97.565**	94.333	3.232
MDPI3_750	750	**97.799**	96.980	0.819
MDPI4_750	750	**96.041**	95.166	0.875
MDPI5_750	750	**96.762**	95.455	1.307
MDPI6_750	750	**99.861**	98.651	1.210
MDPI7_750	750	**96.545**	95.178	1.397
MDPI8_750	750	**96.727**	95.401	1.326
MDPI9_750	750	**98.058**	97.394	0.664
MDPI10_750	750	**100.060**	100.005	0.055

Instance	N	RLTS	DQNTS	Obj. Gap
MDPI1_1000	1000	**119.174**	117.273	1.901
MDPI2_1000	1000	**113.525**	113.417	0.108
MDPI3_1000	1000	**115.139**	113.680	1.459
MDPI4_1000	1000	**111.150**	109.882	1.268
MDPI5_1000	1000	**112.723**	112.723	0.000
MDPI6_1000	1000	**113.199**	110.309	2.810
MDPI7_1000	1000	**111.556**	109.730	1.826
MDPI8_1000	1000	**111.263**	110.369	0.894
MDPI9_1000	1000	**115.959**	114.812	1.147
MDPI10_1000	1000	**114.731**	113.717	1.014

Instance	N	RLTS	DQNTS	Obj. Gap
MDPI1_5000	5000	240.159	**245.624**	5.465
MDPI2_5000	5000	241.827	**245.669**	3.842
MDPI3_5000	5000	240.890	**245.858**	4.968
MDPI4_5000	5000	240.997	**248.330**	7.333
MDPI5_5000	5000	242.480	**245.765**	3.315
MDPI6_5000	5000	240.329	**246.188**	5.859
MDPI7_5000	5000	242.820	**245.399**	2.579
MDPI8_5000	5000	241.195	**244.498**	3.303
MDPI9_5000	5000	**239.761**	211.580	28.181
MDPI10_5000	5000	243.474	**247.482**	4.008

References

[1] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665*, 2017.

[2] Dieudonné Nijimbere, Songzheng Zhao, Xunhao Gu, Moses Olabhele Esangbedo, and Nyiribakwe Dominique. Tabu search guided by reinforcement learning for the max-mean dispersion problem. *Journal of Industrial & Management Optimization*, 2020.