Automatic Optimal Multi-Agent Path Finding Algorithm Selector (Student Abstract)

Jingyao Ren, Vikraman Sathiyanarayanan, Eric Ewing, Baskin Senbaslar, Nora Ayanian

Department of Computer Science
University of Southern California
{jingyaor, sathiyan, ericewin, baskin.senbaslar, ayanian}@usc.edu

Abstract

Solving Multi-Agent Path Finding (MAPF) problems optimally is known to be NP-Hard for both make-span and total arrival time minimization. Many algorithms have been developed to solve MAPF problems optimally and they all have different strengths and weaknesses. There is no dominating MAPF algorithm that works well in all types of problems and no standard guidelines for when to use which algorithm. Therefore, there is a need for developing an automatic algorithm selector that suggests the best optimal algorithm to use given a MAPF problem instance. We propose a model based on convolutions and inception modules by treating the input MAPF instance as an image. We further show that techniques such as single-agent shortest path annotation and graph embedding are very effective for improving training quality. We evaluate our model and show that it outperforms all individual algorithms in its portfolio, as well as an existing state-of-the-art MAPF algorithm selector.

Introduction

Multi-Agent Path Finding (MAPF) is the problem of finding collision free paths for a team of agents traveling from various start locations to goal locations on a map. Solving the MAPF problem optimally is known to be NP-Hard for both make-span and total arrival time minimization (Yu and LaValle 2013). Many types of optimal MAPF algorithms and their variants have been proposed, such as Conflict Based Search (CBS) (Sharon et al. 2015), a method based on branch-and-cut-and-price (BCP) (Lam et al. 2019), and a boolean satisfiability based solver (SAT) (Surynek et al. 2016). However, there is no dominating optimal algorithm that performs well in all types of MAPF problem instances. For example, CBS performs well on maps with narrow corridors, but slows significantly on maps with open spaces (Sharon et al. 2015). It has been shown that if a MAPF problem can be decomposed into multiple disjoint sub-problems, using proper algorithms for each subproblem will be faster than just using one algorithm (Svancara and Bartak 2019). The difficulty of hand-picking the best MAPF algorithm for a particular instance necessitates the development of an algorithm selector which automatically selects the best MAPF algorithm.

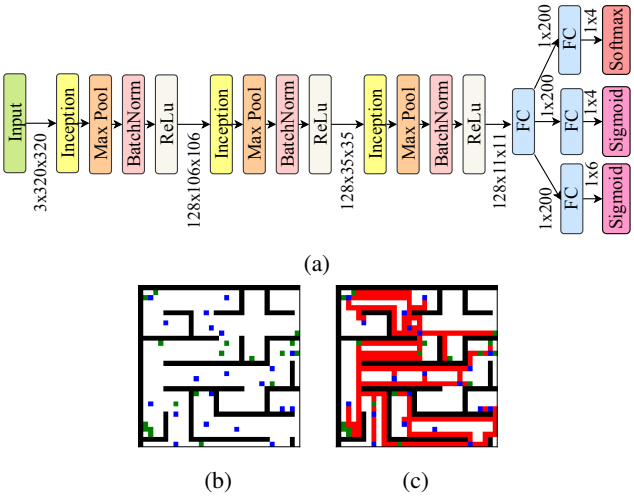

Figure 1: (a). The CNN architecture of our model. Annotating the map image with (b) start and goal locations and (c) single-agent shortest paths.

Automatic MAPF algorithm selectors have not been well studied in the literature. Sigurdson et al. first proposed a classification model based on a convolutional neural network (CNN) (Sigurdson et al. 2019), but their neural network design architecture was not refined for MAPF problems and has limited performance. Kaduri et al. proposed another classification model based on a tree-based learning algorithm (Kaduri, Boyarski, and Stern 2020); this work requires hand-crafted MAPF features (e.g., number of agents, obstacle density), which may impair performance should some important features of an instance not be included.

In this paper, we propose a novel MAPF algorithm selector based on a convolutional neural network (CNN) with inception modules (Szegedy et al. 2015) that outperforms all existing MAPF algorithm selectors. We propose two methods for improving the training quality: annotating the MAPF instances with single-agent shortest path for each agent, and using graph embedding techniques. We also introduce a more selective and up-to-date algorithm portfolio that includes optimization, satisfiability, and search based algorithms (namely BCP, SAT, CBS and CBS-H (Li et al. 2019)).

Algorithm	Accuracy	Coverage	Runtime
BCP	**0.5294**	**0.91**	2256
CBS	0.1888	0.41	7714
CBS-H	0.2810	0.90	**2211**
SAT	0.0008	0.38	8548
XGBoost Cl	0.6711	0.95	1694
G2V	0.7130	0.95	1548
MAPFAST	**0.7689**	**0.97**	**1339**
Oracle	1.0	1.0	917

Table 1: Simulation results

Model

In this work, we have modeled algorithm selection as a classification task using a CNN, and implemented it in Python using the PyTorch package. Our model takes as input an image representation of the MAPF instance and returns a prediction of the fastest algorithm. Inception modules are used to improve training speed and allow for a much deeper network. The architecture of our CNN is three inception modules followed by a fully connected layer and three output layers (Fig. 1a). Each inception module is followed by a max-pool layer and a batch normalization layer. The input for the CNN is a 320×320 map image annotated with optimal paths for each agent without considering other agents (Fig. 1c). Based on our experiments, using maps annotated with single-agent shortest paths rather than only start/goal locations (Fig. 1b) significantly improves training quality.

Simulation Results

We used three metrics to evaluate our model. *Accuracy* gives the proportion of instances that the fastest algorithm is correctly selected, or, for a single algorithm, the proportion of instances it is the fastest algorithm. *Coverage* is the proportion of the instances that an algorithm successfully solved. *Runtime* is the overall time taken for the predicted algorithm, in minutes, to solve all the problems in the test set. A default value of 5 minutes was added to runtime when an algorithm didn't solve the instance within the 5 minute time limit.

Our algorithm portfolio has four algorithms: BCP, CBS, CBS-H and SAT. This covers optimization-, search- and satisfiability-based approaches. Table 1 shows the simulation results from testing 2484 MAPF instances from the benchmark set (Stern et al. 2019). The first four rows are where a single algorithm is used on all the input instances. It can be seen that BCP and CBS-H were successful in solving 90% of the input instances. However, even for the best individual algorithm, BCP, the accuracy was only 53%, meaning that it is the fastest algorithm for 53% of the instances. Choosing only the BCP algorithm would take more than twice as long as choosing the best performing algorithm for each instance. This further justifies the claim that there is no dominating optimal MAPF algorithm. The second part of the table shows the comparison of a state-of-the-art MAPF algorithm selector, *XGBoost Cl* (Kaduri, Boyarski, and Stern 2020), and our models. The *Oracle* row gives the analysis for always using the fastest algorithm. Our CNN model, named MAPFAST, successfully predicted the fastest algorithm for 77% of the input instances and had coverage of 97%, outperforming XGBoost Cl, which had 67% accuracy and 95% coverage. We also tried transforming the map image properties into vectors using a graph embedding framework called *Graph2Vec* (Narayanan et al. 2017). Our model that used graph embeddings, named G2V, had performance comparable to MAPFAST, with 71% accuracy and 96% coverage, also outperforming XGBoost Cl. Different from XGBoost Cl and MAPFAST, G2V is trained only using the single-agent shortest paths without knowing the topology of instance map. G2V can also handle different map sizes without re-training, which is more flexible than the CNN-based approaches. The total runtime for the algorithms predicted by our models are significantly less than using an individual algorithm all the time. Our models also have a remarkable improvement of accuracy compared to all of the individual algorithm, which further justifies our design.

References

Kaduri, O.; Boyarski, E.; and Stern, R. 2020. Algorithm Selection for Optimal Multi-Agent Pathfinding. In *ICAPS*, volume 30, 161–165.

Lam, E.; Le Bodic, P.; Harabor, D. D.; and Stuckey, P. J. 2019. Branch-and-Cut-and-Price for Multi-Agent Pathfinding. In *IJCAI*, 1289–1296.

Li, J.; Felner, A.; Boyarski, E.; Ma, H.; and Koenig, S. 2019. Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search. In *IJCAI*, 442–449.

Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y.; and Jaiswal, S. 2017. Graph2Vec: Learning Distributed Representations of Graphs. *arXiv preprint arXiv:1707.05005* .

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence* 219: 40–66.

Sigurdson, D.; Bulitko, V.; Koenig, S.; Hernandez, C.; and Yeoh, W. 2019. Automatic Algorithm Selection in Multi-Agent Pathfinding. *arXiv preprint arXiv:1906.03992* .

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Bartak, R. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Symposium on Combinatorial Search* 151–158.

Surynek, P.; Felner, A.; Stern, R.; and Boyarski, E. 2016. Efficient SAT Approach to Multi-Agent Path Finding Under the Sum of Costs Objective. In *ECAI*, 810–818.

Svancara, J.; and Bartak, R. 2019. Combining Strengths of Optimal Multi-Agent Path Finding Algorithms. In *ICAART*, 226–231.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going Deeper with Convolutions. In *CVPR*, 1–9.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *AAAI*, 1444–1449.