GPU-accelerated Algorithms on solving Stochastic Shortest Path Problems Proposal

Duo Zhou
Department of Industrial & Enterprise Engineering
University of Illinois Urbana-Champaign
Urbana, IL 61801
duozhou2@illinois.edu

Yilan Jiang
Department of Industrial & Enterprise Engineering
University of Illinois Urbana-Champaign
Urbana, IL 61801
yilanj2@illinois.edu

Tharunkumar Amirthalingam
Department of Industrial & Enterprise Engineering
University of Illinois Urbana-Champaign
Urbana, IL 61801
ta9@illinois.edu

Abstract

The abstract paragraph should be indented ½ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

Stochastic shortest path is a problem in which we need to find the shortest path from a given start node to a goal node in a graph, where the edge weights are not deterministic but instead are random variables. This problem is commonly used in many applications, such as robotics, transportation, and network routing.

Regular shortest path algorithms, such as Dijkstra's algorithm or the A* algorithm, assume that the edge weights in a graph are deterministic and known in advance. These algorithms work well when the edge weights are fixed and do not change over time. However, in many real-world applications, the edge weights are not fixed but instead are subject to randomness and uncertainty. For example, in transportation networks, the travel time between two locations can vary depending on traffic conditions, weather, accidents, etc. Similarly, in robotic navigation, the cost of moving from one location to another can depend on sensor readings, terrain features, obstacles, etc. In such scenarios, regular shortest path algorithms may not be appropriate as they do not take into account the randomness and uncertainty of the edge weights. Stochastic shortest path algorithms, on the other hand, explicitly model the probabilistic nature of the edge weights and aim to find the path with the lowest expected cost.

Preprint. Under review.

The stochastic shortest path problem is a challenging problem in stochastic optimization, and its solution requires a combination of mathematical and computational techniques. The need for using GPUs, particularly CUDA, to speed up the stochastic shortest path algorithm depends on various factors such as the size of the graph, the complexity of the edge weight distributions, and the available computing resources. In general, stochastic shortest path algorithms involve computations with matrices and vectors, which can be computationally intensive for large graphs. GPUs, with their massively parallel architecture and high memory bandwidth, can accelerate these computations significantly, leading to faster computation times. Moreover, if the edge weight distributions are complex and involve high-dimensional probability distributions, such as multivariate normal or mixture distributions, then the computations involved in the algorithm may be even more demanding. In such cases, GPUs can provide significant speedups compared to CPU-based implementations.

2 Literature Review

There have been plenty of algorithms innovated to resolve the general shortest path problem in the past decades. Gallo and Pallottino discussed eight algorithms for the shortest path problem and compared their performances with various data structures [1]. To further increase the computational efficiency, Crauser et alṗroposed a parallelized version of the Dijkstra's algorithm [2]. Even though the shortest path problem can be applied to many areas like road networks and social media, it is less generic to be applied to some other real life problems such as robotics, autonomous driving, and network routing.

Bertsekas and Tsitsiklis generalized the shortest path problem and first introduced the idea of stochastic shortest path (SSP) problem in 1991 [3]. They further extended the corresponding theory of Markovian decision problems by removing the assumptions that cost can either be all positive or all negative. Building on top of that, Polychronopoulos and Tsitsiklis developed a dynamic programming algorithm to resolve a SSP problem to devise a policy that leads from an origin to a destination with minimal expected cost (4). Further works have been done to expand the ecosystem of SSP problem by proposing new concepts and frameworks [5, 6]. In addition, SSP is defined as an instance of a Markov Decision Process.

On Markov Decision Processes with Parallel Algorithm, Archibald, T. W. et alİooked at serial value iteration algorithms for Markov decision processes and develops efficient parallel alternatives [7]. Jóhannsson, Á.Þ. et alintroduced two CUDA-based implementations of the Value Iteration algorithm: Block Divided Iteration and Result Divided Iteration and showed a substantial performance improvement for the parallel algorithms compared to a sequential implementation on a CPU [12]. Ruiz and Hernandez formulated a MDP solver based on the Value Iteration algorithm that uses matrix multiplications. This allows us to leverage GPUs to produce interactive obstacle-free paths in the form of an Optimal Policy [10].

Ortega-Arranz, H. et alṗresented a GPU SSSP algorithm implementation significantly sped up the computation of the SSSP. They also enhanced the GPU algorithm implementation using proper choice of threadblock size and the modification of the GPU L1 cache memory state of NVIDIA devices [11]. Sapio et alintroduced two new MDP solvers for embedded systems: Sparse Value Iteration (SVI), which operates on small, single-threaded CPU platforms using sparse matrix methods, and Sparse Parallel Value Iteration (SPVI), which takes advantage of embedded graphics processing units (GPUs) to increase performance on more advanced embedded systems [8]. Steimle et alḋeveloped exact and fast approximation methods with error bounds for Multi-model Markov decision process (MMDP), which generates a single policy maximizing performance over all models. This approach allows the decision maker to trade-off conflicting data sources while creating a policy of the same complexity for models that consider only one data source [9].

3 Proposed work

3.1 Tasks

In Task 1, we will spend sometime to explore the mathematical theories as well as algorithms relevant to stochastic shortest path and markov decision processes in greater details. By diving into these

Table 1: Milestons

Time	Tasks
Mar. 9	Proposal Investigation
Mar. 23	Review of Literature & Algorithm Design
Mar. 28	Finish up Miderm Report
Apr. 13	Mathematical proof of Algorithm
Apr. 23	Experimental Design & Validation
Apr. 25	Final Report Completion

topics, we will be able to better understand the underlying principles that govern these processes and apply this knowledge to real-world situations.

For Task 2, in order to solve the stochastic shortest path problem, we can reproduce various existing algorithms. These algorithms can be implemented using a CPU as a baseline scenario. By utilizing these techniques, we can efficiently identify the optimal path in a given graph, even when the graph has stochasticity. Additionally, we may consider implementing these algorithms on GPUs especially using CUDA in order to further optimize the solution.

3.2 Data

We will evaluate our method on a list of infinite-horizon Multi-model Markov Decision Processes (MMDPs). These MMDPs are test instances generated by using the codes provided by University of Michigan-Deep Blue Data. We will use their codes to generate three sets of MMDPs, each corresponding to different medical decision-making problems. These problems include the initiation of HIV therapy, a machine maintenance problem, and a set of randomly constructed instances. A more detailed description of the data under each problem can be found online.

`https://deepblue.lib.umich.edu/data/concern/data_sets/t722h899p`

3.3 Milestons

Referring to the schedule Table 1, our milestone is shown above.

3.4 Testing and Expected results

We will choose CUDA enabled NVIDIA GPU and C++ to realize the paralleled computations and use python to take care of all other programming works including data preprocessing and pipeline construction. We will test our method compared to the baseline model using CPU counterparts in terms of speed. It is important to choose the appropriate method or algorithm for the specific problem at hand and to carefully analyze the results to ensure that the solution is accurate and effective.

References

[1] Gallo, G., & Pallottino, S. (1988). Shortest path algorithms. *Annals of operations research* **13**(1): 1-79.

[2] Crauser, A., Mehlhorn, K., Meyer, U., & Sanders, P. (1998). A parallelization of Dijkstra's shortest path algorithm. *In Mathematical Foundatio s of Computer Science 1998: 23rd International Symposium MFCS'98 Brno, Czech Republic, August 24-28, 1998 Proceedings* **23**(pp. 722-731). Springer Berlin Heidelberg.

[3] Bertsekas, D. P., & Tsitsiklis, J. N. (1991). An analysis of stochastic shortest path problems. *Mathematics of Operations Research* **16**(3): 580-595.

[4] Polychronopoulos, G. H., & Tsitsiklis, J. N. (1996). Stochastic shortest path problems with recourse. *Networks: An International Journal* **27**(2): 133-143.

[5] Ji, X. (2005). Models and algorithm for stochastic shortest path problem. *Applied Mathematics and Computation* **170**(1): 503-514.

[6] Guillot, M., & Stauffer, G. (2020). The stochastic shortest path problem: a polyhedral combinatorics perspective. *European Journal of Operational Research* **285**(1): 148-158.

[7] Archibald, T. W., McKinnon, K. I. M., & Thomas, L. C. (1993). Serial and parallel value iteration algorithms for discounted Markov decision processes. *European journal of operational research* **67**(2): 188-203.

[8] Sapio, A., Bhattacharyya, S. S., & Wolf, M. (2020, July). Efficient model solving for Markov decision processes. *In 2020 IEEE Symposium on Computers and Communications (ISCC)*(pp. 1-5). IEEE.

[9] Steimle, L. N., Kaufman, D. L., & Denton, B. T. (2021). Multi-model Markov decision processes. *IISE Transactions* **53**(10): 1124-1139.

[10] Ruiz, S., & Hernández, B. (2015, October). A parallel solver for Markov decision process in crowd simulations. *In 2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI)*(pp. 107-116). IEEE.

[11] Ortega-Arranz, H., Torres, Y., Gonzalez-Escribano, A., & Llanos, D. R. (2015). Comprehensive evaluation of a new GPU-based approach to the shortest path problem. *International Journal of Parallel Programming* **43**(5), 918-938.

[12] Jóhannsson, Á.Þ. (2009). GPU-based Markov decision process solver.