Shortest Path First Algorithm Implementation Analysis

Research and Analysis

CONTACTS

|  |  |
| --- | --- |
| ROLE | Name; Alias |
| Program User | Max Zilberman (mazilber) |
| Developer | Danny Garber |

# Executive Summary

[Guidelines](https://microsoft.sharepoint.com/teams/osg_threshold_specs/SitePages/ExecSummary.aspx)

When dealing with managing baggage handling at Hartsfield-Jackson Atlanta International airport, one of the largest airports in the world, and considering the 192 departing gates in the airport’s vast area we are given a directed acyclic graph (DAG) with dynamic edge costs. The costs are dynamic because of the stochastic nature of the multiple frequent events that will drive these costs weight in a very rapid fashion. Additionally, the upper bound for getting the optimum shortest path is no more than 2 seconds.

The goal of this research is to analyze various techniques of implementing the shortest path algorithms running them on various hardware configurations and to determine the uniform best cost search algorithm for the given upper bound constrain.

# Terminology and Definitions

|  |  |  |
| --- | --- | --- |
| Term | Definition | Reference |
| SPF | Shortest Path First algorithm for finding the shortest paths between nodes in a graph |  |
| Node | A given location on the graph from which a tug agent can depart or arrive. An example: a gate, storage facility, concourse check-in terminal. |  |
| DAG | A directed acyclic graph |  |
| Weight | A numeric representation of the cost associated with arriving from one node to another |  |
| Floyd-Warshall Algoritthm | Floyd-Warshall Algorithm is an algorithm based on dynamic programming technique to compute the shortest path between all pair of nodes in a graph | <https://en.wikipedia.org/wiki/Floyd–Warshall_algorithm> |
| Dijkstra’s Algorithm\* (modified) | Dijkstra’s Algorithm is an algorithm for finding the shortest paths between nodes in a graph when few edges (sparse) are present | <https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm> |
| FPGA | A field-programmable gate array is an integrated circuit designed to be configurable using a hardware description language (HDL) used to perform line-rate computation |  |
| MST | Minimum Spanning Tree algorithm is an a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. | <https://en.wikipedia.org/wiki/Minimum_spanning_tree> |
| Kruskal’s Algorithm | Kruskal's algorithm is a minimum-spanning-tree algorithm which finds an edge of the least possible weight that connects any two trees in the forest. | <https://en.wikipedia.org/wiki/Kruskal%27s_algorithm> |
| Boruvka’s algorithm | Boruvka’s algorithm is an algorithm for finding a minimum spanning tree in a graph for which all edge weights are distinct, or a minimum spanning forest in the case of a graph that is not connected | <https://en.wikipedia.org/wiki/Borůvka%27s_algorithm> |
| KKT | Karger Klein Tarjan Expected Linear Time Minimum Spanning Tree algorithm |  |

# Goals & Non-Goals

[Guidelines](https://microsoft.sharepoint.com/teams/osg_threshold_specs/SitePages/Goals_NonGoals.aspx)

## Goals & Measures

|  |  |  |  |
| --- | --- | --- | --- |
|  | Goal | Measures | Priority |
| 1 | Find the best uniformed shortest path search algorithm for the given problem and constrains | Maximum # of nodes computed in 2 seconds | **P1** |
| 2 | Formulate the best cost-effective hardware configuration and specs for running the computation | Cost vs Performance | **P1** |
| 3 | Analyze various programming language implementation techniques and provide the recommendation for the most effective one | Maximum # of nodes computed in 2 seconds | **P1** |

## Non-Goals

1. Provide a generic SPF algorithm implementation
2. Provide a solution for the graph containing negative cycles
3. Provide a solution for the graph containing negative weights

Playbook Scenario

Tasks:

* 1. Write a pseudocode for each SPF algorithm
  2. Implement each algorithm in C++, C# and Python
  3. Measure each solution within the given constrains
  4. Improve each implementation solution by introducing a hardware multi-core parallelism and graphics processing unit (GPU) computation
  5. Re-take the measurements for each solution
  6. Draw the conclusion and report the findings

Algorithms will be tested:

1. Floyd-Warshall
2. Dijkstra’s algorithm (modified for all nodes path)
3. Kruskal’s algorithm (for MST)
4. Randomized KKT algorithm
5. Optional: research and investigate use of FPGA and modify the computation details

Test Machine Hardware Specifications:

1. Intel® Core™ i9-9960X CPU @4.1GHz
2. 64 GB DDR-4200 RAM
3. Nvidia GeForce 1080 Ti GPU
4. M.2 PCIe NVMe 2TB RAID 0 Internal SSD
5. Windows 10 64-bit