CS-205 Project:Evacuation plan for a building

Sonu Mehta, Tomas Gudmundsson, Dali Moghimi

April 2017

1 Introduction

This interim progress report describes the design and analysis of an evacuation strategy for an urban multi-tiered structures such as skyscrapers. Additional details will be included later.

2 Problem Description

The objective is to find the most efficient and expedient routes to the structure's exits for evacuees in an emergency situation such that each person is directed to one of the nearest exits on each floor and then led down the stairwells to the ground floor, taking into account congestion at the floor exits as evacuees converge during their spree.

Concretely, we start with the floor plan of a structure. This plan is converted into a weighted, undirected graph with nodes equal to the sum of the number of occupants and exits available for each particular floor. The graph is a connected graph with edge weights representing the distance between two nodes. In the case of an emergency evacuation, the goal is to assign to each person the nearest exit whilst taking into account the congestion that may occur at these exits. Each exit has a specified capacity that must not be exceeded in order to safely evacuate the occupants. The algorithm is expected to output the optimal or near optimal assignment of evacuees to the exits such that the total cost is minimized. The total cost in this problem is defined as the sum of the distances of each person from their assigned exit and the penalty incurred for exceeding a the capacity for a particular exit.

3 Algorithm

An adjacency matrix is utilized in order to implement the graph abstraction of the floor plan of the structure. This graph is a randomly generated, connected, undirected, and weighted graph with a few random nodes assigned as exits. For each exit, we use Dijkstra's algorithm to find the shortest distances to each node (representing an occupant). Once we have the distance to each exit for every occupant, we assign the nearest exit to each person and calculate the cost of the assignment. For now, we will assume that the congestion can only occur at the exits and that the edges do not have flow constraints, i.e., any number of occupants can pass freely along an edge at any given point in time.

Once the assignments to the nearest exits for each occupant are complete, we use simulated annealing to find the minimum cost configuration. After each iteration of the SA algorithm, we generate a new configuration by changing the exit assignment of one the occupants at random while the temperature is gradually reduced to zero. If the new assignment has a lower cost than the older assignment, the algorithm will accept the new configuration and if the cost is higher, the algorithm will only accept the new configuration with probability $= exp((newCost - oldCost))/T$ where T is the temperature. The algorithm is expected to give a good approximation of the global minimum cost configuration.

4 Parallel implementation of algorithm

The above algorithm is sequential and calculates the shorted distance using Dijkstra one by one. Since Dijkstra from one exit is not affected by another exit, we parallelize the algorithm by using openmp to calculate the shortest distance from all exits in parallel. This speeds up the process of calculation of shortest distance. Similarly, for simulated annealing, instead of starting with 1 initial configuration, we start with multiple initial configurations in parallel and take the best solution at the end. Since Dijkstra's algorithm applied to one exit is independent of the other exits, we parallelize the algorithm using OpenMP as our framework in order to calculate the shortest distance paths from all exits in parallel. This speeds up the process of calculation of shortest distance by a factor of roughly three. Similarly, for simulated annealing, instead of starting with 1

initial configuration, we start with multiple initial configurations in parallel and take the best solution at the end.

The above algorithm is sequential and calculates the shorted distance using Dijkstra one by one. Since Dijkstra from one exit is not affected by another exit, we parallelize the algorithm by using openmp to calculate the shortest distance from all exits in parallel. This speeds up the process of calculation of shortest distance. Similarly, for simulated annealing, instead of starting with 1 initial configuration, we start with multiple initial configurations in parallel and take the best solution at the end.

5 Next Steps:

The first iteration of the evacuation algorithm is working. It is expected to find the optimal (or near optimal) solution using simulated annealing.
In proceeding further with the project, we would like to compare the implementation of Dijkstra with the Shoshan-Zwick algorithm or a similar matrix multiplication algorithm in order to compute the shortest paths. All-Pairs Shortest Paths with matrix multiplication is ripe for the application of at least a few parallelization paradigms. If time permits, we may add additional constraints in order to increase the complexity of the problem. Possibilities may include adding edge flow capacities and varying the rates of traversal for occupants.