Rochester Institute of Technology

Project Proposal

Algorithms for the Widest Path Problem

Venkat Sasank Donavalli

Committee Chair: Dr. Edith Hemaspaandra
Reader:
Department of Computer Science
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

July 29, 2013
Contents

1 Introduction .............................................. 3

2 Background and Related Work ......................................... 3
   2.1 Problem Description ........................................ 3
   2.2 Known Algorithms .......................................... 4
   2.3 Graph Processing using Distributed Algorithms ...................... 5
   2.4 Applications .................................................. 6

3 Problem Statement ............................................... 6

4 Proposed Work .................................................. 6

5 Deliverables ...................................................... 7

6 Project Outline and Schedule ...................................... 7
   6.1 Project Outline ............................................... 7
   6.2 Timeline ........................................................ 8

List of Figures

1 Example Graph .................................................. 3
Abstract

The widest path problem is a graph problem which is defined as finding the most optimal path between two vertices in a weighted graph which has the maximum possible weight for the most minimum weighted edge in the path. The widest path problem has applications in, for example, max flows, network routing, digital imaging, and voting theory. The widest path problem is related to the shortest path problem and algorithms known to solve this problem can be modified to solve the widest path problem. However, a different and more complicated approach is often needed to get optimal algorithms. Many of these algorithms exist only in theoretical papers, and so a major contribution of this project will be to explain these algorithms in an accessible way.

In this project I propose to study, compare, explain, and implement the algorithms known for solving the widest path problem and to investigate the various applications for this problem. I also intend to come up with an optimized distributed algorithm to help find the widest path for a highly dense graph.
1 Introduction

Graphs are usually defined as a set of vertices which are interconnected by a set of edges. The set of edges can either be unidirectional or directional. In the former case the graph is known as a undirected graph and the latter is known as a directed graph. Also when the edges are associated with a quantitative value the graph is known as a weighted graph. Graphs have been used to represent and solve problems in several different application areas. One of the most widely used graph type problem include finding optimal paths in a given graph under some given constraints.

Widest path problem[1] is also commonly referred to as the bottleneck shortest path problem and the maximum capacity route problem. It is a path finding graph problem which has been used for several different applications such as finding max flows, network routing and digital imaging. It also used in Schulze's method[8] for finding the winner of a multiway election.

This study focuses at the variants of the widest path problem and the different algorithmic approaches to solving this problem. We also look to investigate a distributed algorithmic approach for solving it for highly dense graphs.

![Example Graph](image1)

Figure 1: Example Graph

2 Background and Related Work

2.1 Problem Description

We start with a formal definition of the widest path problem. Let the given graph be defined as $G(V, E)$, where $V$ is the set of vertices and $E$ is the set of edges. For every edge $e \in E$ we define $w_e$ as the weight of that edge. Using this we can now define the widest path problem as finding the path $P$ in the graph $G(V, E)$ between vertex $i$ and vertex $j$ (where $i, j \in V$) such that weight $W$: $\min(w_x : x \in E \& x \in P)$ is maximum. The value $W$ is also referred sometimes as the bandwidth of the path. Hence the widest path will also have
the maximum bandwidth in the graph.

In order to understand this better we describe the problem using the following example. The figure 1 represents a weighted directed graph. We are interested in finding the widest path from vertex $A$ to vertex $F$. Using our earlier definition we can see that the widest path for this graph would be the path $A \rightarrow C \rightarrow D \rightarrow F$. Also the bandwidth for this path is equal to the min weighted edge in this path which is equal to 11.

The widest path can be determined for both directed and undirected graphs. Some of the variants for the widest path problem are single source single destination, single source all destinations and all-pairs.

### 2.2 Known Algorithms

Due to its similarity with single source shortest path problem there are some well-known algorithmic approaches for solving the widest path problem for directed and undirected graphs. One common approach for solving this problem is to modify the Dijkstra’s algorithm for solving the shortest path problem. Another approach discussed by Kaibel and Peinhardt[5] involves using s-t cuts for solving single source single destination variant of the widest path problem in case of undirected graphs.

The above mentioned approaches are not efficient for solving the all-pairs variant of the widest path problem. One approach for solving the all-pairs widest path problem is modifying the Floyd-Warshall’s algorithm[4]. Floyd Warshall’s algorithm is used to solve the all-pairs shortest path problem by comparing the distance of all possible paths in the given graph for each pair of vertices. The path having the minimum distance is selected as the shortest path for the pair of vertices. In order to use this algorithm to solve the widest path problem we need to compare using the bandwidth instead of using the distance. Also among all the paths possible for a pair of vertices the path having the maximum bandwidth is chosen as the widest path for that pair of vertices.

The modified Floyd-Warshall’s algorithm for the widest path problem is shown below. We have defined two 2D arrays, one for storing the bandwidth of each pair of vertices and another for generating the path corresponding to the bandwidth. The function $\text{FloydWarshall\_AllPairs\_WidestPathProblem}$ is used for finding the bandwidth for all pairs and the function $\text{Trace\_Path}$ is used to reconstruct the widest path for a given pair of vertices.

**Initialize** bandwidth as an array of size $|V| \times |V|$ to be $-\infty$

**Initialize** next as an array of size $|V| \times |V|$ to be $null$

```plaintext
function FloydWarshall\_AllPairs\_WidestPathProblem (G(V,E))
    for each vertex $v$ in $V$ do
        bandwidth[$v$][$v$] ← 0
    end for
    for each edge $(u, v)$ in $E$ do
        bandwidth[$u$][$v$] ← $w(u, v)$
    end for
    for $k$ from 1 to $|V|$ do
        for $i$ from 1 to $|V|$ do
            for $j$ from 1 to $|V|$ do
                if min(bandwidth[$i$][$k$], bandwidth[$k$][$j$]) > bandwidth[$i$][$j$] then
```
bandwidth[i][j] ← min(bandwidth[i][k], bandwidth[k][j])
next[i][j] ← k
end if
end for
end for
end for
end function

function Trace_Path(i, j)
    if bandwidth[i][j] == −∞ then
        return “There does not exist any path from i to j”
    end if
    k ← next[i][j]
    if k == null then
        return “”
    else
        return Trace_Path(i, j) + k + Trace_Path(k, j)
    end if
end function

This above modified Floyd-Warshall’s algorithm has a running time of O(n^3) where n corresponds to |V|. However all-pairs widest path problem can be solved in sub cubic time[11] using certain algorithms. One such algorithm is the maxmin matrix multiplication algorithm proposed by Duan and Pettie[2]. Later on we will be covering these and other algorithms[9] in detail. Most of these algorithms have been mentioned theoretically and lack proper implementation. As a part of this project we will be implementing most of these algorithms for the first time. The implementations would also help us do a better comparison between the algorithms.

2.3 Graph Processing using Distributed Algorithms

In today’s information centric world Graphs are a key data structure used for representing the connection between the different pieces of information. As such most application areas can have graphs with several million vertices and edges. Social networks and web graphs are two examples were the generated graphs are generally highly dense. Processing such graphs using traditional algorithms can be expensive and would require having very large memory systems. Also it becomes hard to scale with the increase in the graph density. In order to address these issues we propose using distributed algorithms.

Distributed algorithms generally help us to segment an existing large problem into a set of smaller independent problems which can then be processed in parallel on different machines. The final solution is obtained by combining the results of the smaller problems. These steps in a distributed algorithm are commonly represented using the MapReduce[6] model.

It can be difficult to come up with scalable distributed algorithms for graphs. One commonly used technique is graph partitioning[7] which we would be discussing in details later on.
### 2.4 Applications

One of the most widely known applications of the widest path problem is in the Schulze’s method\[8\] for finding a single winner in a multi-way voting system. The Schulze’s method allows the following voting guidelines.

- The voters provide an ordered preference list of the candidates.
- Voters can assign the same preference to multiple candidates.
- It is not required to use consecutive numbers to define to candidate preference order.
- Voters are not required to rank all candidates.

Using the final votes Schulze’s method constructs a weighted directed graph where the vertices corresponds to the candidates and directed edges signify the preference between a pair of candidates. A directed edge from candidate \( C_1 \) to candidate \( C_2 \) implies that overall voters prefer candidate \( C_1 \) over \( C_2 \). In this case the weight for the edge between \( C_1 \) and \( C_2 \) represents the number of voters who ranked \( C_1 \) higher than \( C_2 \). The next step in Schulze’s method involves finding the strength of the strongest path between every pair of candidates. The strength of the strongest path from a candidate \( C_i \) to candidate \( C_j \) corresponds to the bandwidth of the widest path from \( C_i \) to \( C_j \). Hence Schulze’s method applies all-pair widest path problem in order to find the final winning candidate. Schulze’s method solves the all-pair widest path problem by modifying the Floyd-Warshall algorithm mentioned earlier.

Another application of the widest path problem is in mosaicking of digital photographic maps\[3\] in order to produce a larger map. Mosaicking is a technique for combing multiples images into a single composite image. Besides that the widest path problem also finds application in metabolic path analysis\[10\] for living organisms. Metabolic path can be analyzed by constructing a metabolic network based on the stoichiometry of the reactions. The dominant paths in the network are determined by the limiting step in the reaction path. This corresponds to the bandwidth of the widest path. Hence metabolic path analysis uses the solution of the single source widest path problem. We will be discussing these applications in detail later on.

### 3 Problem Statement

Study and compare the different algorithms known for solving the widest path problem and also come up with a distributed algorithmic approach for solving the problem. The final output would consist of implementation of all the algorithms including the distributed algorithm. The implementations would be for both directed and undirected graphs. It will also cover the variants for single source/single destination, single source all destinations and all pairs. It would also include a comparison report of the different algorithms.

### 4 Proposed Work

The focus of this project lies in studying and comparing the different algorithms known for solving the widest path problem. This means implementing the algorithms and comparing them on a set of characteristics such as time and space complexity. Another keen area of this project involves coming up with a distributed algorithm to solve this problem and evaluating its performance against a highly dense graph.
The project work would require performing various experiments in order to compare the various algorithms. Java would be used as the primary coding language for the implementation. Also some existing libraries may be used to simplify testing.

The final results obtained will be recorded in the project report. The source code will be also submitted separately along with the project report.

5 Deliverables

Below are deliverables which would be submitted upon the completion of the project.

- Project report summarizing the overall work done for the project including the comparisons, results and the final analysis.
- Completed source code of all the algorithm implementations
- A comprehensive documentation for running the source code and reproducing the submitted results
- Final project defense presentation

6 Project Outline and Schedule

6.1 Project Outline

The following is the expected Project Outline.

- Abstract
- Introduction
- Background and Related work
- Survey of Known Algorithms
- Solving using a Distributed Algorithm
- Experiments
- Comparison Results
- Final Analysis
- Bibliography
6.2 Timeline

The following is a tentative schedule for the project completion.

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>July 2013</td>
<td>Background Study</td>
</tr>
<tr>
<td>August 2013 - September 2013</td>
<td>Literature Review</td>
</tr>
<tr>
<td>October 2013</td>
<td>Implementation, Experiments and Results</td>
</tr>
<tr>
<td>November 2013</td>
<td>Report Work</td>
</tr>
<tr>
<td>December 2013</td>
<td>Project defense</td>
</tr>
</tbody>
</table>

REFERENCES


