The Maximum-Flow problem in undirected graphs

A Project Report

submitted by

KARTHIK ABINAV S

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2014

THESIS CERTIFICATE

This is to certify that the thesis entitled **The Maximum-Flow problem in undirected graphs**, submitted by **Karthik Abinav S**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. N. S. Narayanaswamy
Research Guide
Associate Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

Acknoledgements

ABSTRACT

KEYWORDS: Graph Theory, Maximum Flow, Laplacian solvers

Maximum flow problem has been a very important optimization problem in computer science and mathematics. This problem has a lot of practical relevance. Some of the age-old applications involving maximum flow have been in electrical circuits, water supply networks, etc. With the advent of social media and social networks, this problem has found a newer practical relevance. And since the graphs in these networks are typically very large, researchers are sought after creating faster and more efficient algorithms for this problem.

Some of the classical algorithms to solve this problem are the Ford-Fulkerson's augmenting path algorithm and the Dinic's Algorithm. These algorithms compute the exact value of the maximum flow. It is also fairly straightforward to obtain the optimal flow vector after the termination of the algorithm. The main drawback with this algorithm is that the running time, though polynomial, is very high and is expensive to use in many practical situations. Following these algorithms a series of push-relabel algorithms were devised which had a slightly better running time as compared to the classical algorithms. The series of algorithms terminated with the algorithm by Goldberg-Rao which gave a O($m^{\frac{3}{2}}$)-time algorithm. For extremely large graphs, as in the case of social networks, this algorithm is still far from being practical.

In 2008, the breakthrough result by Spielman and Teng gave an algorithm to solve the Symetric Diagonally Dominant(SDD) system of equations in near-linear time. This work was immediately extended by Koutis-Miller-Peng which gave an efficient algorithm to produce an incremental graph sparsifier. Developement of these techniquies led to the developement of an almost linear time algorithm to the maximum flow problem by Cristiano-Kelner-Madry-Spielman-Teng. This involved looking at the graph as a resistive network, approximating the electrical flow and producing an approximate s-t flow from this approximate electrical flow. This algorithm broke the running time barrier of Golderg-Rao and gave the first almost-linear time algorithm.

In this thesis, we give a survey of the above algorithms for the maximum flow problem. We first start off by giving a brief description of the classical algorithms. We then conclude this by giving a completely recursive specification for the push-relabel algorithm. We then present the required tools from spectral graph theory and linear algebra that is required to understand the almost linear time algorithm. Finally, we present the Cristiano-Kelner-Madry-Spielman-Teng algorithm in detail. We will also present some of the key theorems from the Koutis-Miller-Peng SDD solver. Finally, we give some arguments to justify the requirement of the weight updates and also show some steps that can possibly lead to making the algorithm parallel.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
BFS	Breadth First Search
DFS	Depth First Search
SDD	Symetric Diagonally Dominant

NOTATION

m	The number of edges in a graph G
n	The number of vertices in a graph G
u	A $m * 1$ matrix containing the capacities of the edges
U	This defines the ratio between the highest and lowest capacity in the graph, i.e. $\frac{\max_e u_e}{\min_e u_e}$

CHAPTER 1

INTRODUCTION

Maximum-Flow problem is an age old optmization problem in mathematics and computer science. Hundreds of researchers have investigated this problem and have given some interesting results. This problem has far-reaching applications in almost every field of engineering and sciences. Hence, this problem is of great importance, not just theoretically, but also in real-world applications. Hence, getting better and more efficient algorithms for this problem and its variants has always been the pursuit. Inspite, of having had so much research into this problem, for many years we were far from getting an algorithm that runs in time that is good for practical purposes. This gives an indication of the hardness of this problem.

1.1 Basics

Graph

A *graph* $G(V, E, \rho)$ is defined as a mathematical structure on the set of vertices V and the set of edges E and an adjacency function $\rho : V \times V \rightarrow E$, such that $\rho(a, b) = e$ for $a, b \in V$ and $e \in E$ tells that there exists an edge e between the vertices a and b.

Undirected Graph

An *undirected graph* is a graph where the function ρ is symmetric, i.e. $\rho(a, b) = \rho(b, a)$.

Directed Graph

An *directed graph* is a graph where the function ρ is not necessarily a symmetric function.

Capacitated Graph

A *capacitated graph* is a graph defined as $G(V, E, \rho, \psi)$, where the first three values in the tuple mean the same as before. The ψ in the definition is a function $\psi : E \to \mathbb{R}$, which assigns a real number corresponding to every edge e in the graph. This real number is called the *capacity* of the graph.

s-t Flow

A *s-t flow* is a vector $f \in \mathbb{R}^{|E|}$ such that the following two criteria hold:

- Flow Conservation:
$$\sum_{e \in E^+(v)} f(e) - \sum_{e \in E^-(v)} f(e) = 0 \quad \forall v \in V \setminus \{s, t\}$$
where $E^-(v)$ denotes the set $\{a : \rho(a, v) \text{ is defined}\}$ and $E^+(v)$ denotes the set $\{a : \rho(v, a) \text{ is defined}\}$
- Capacity maintanence:
$$f(e) \leq \psi(e) \quad \forall e \in E$$

Additionally, the *value* of the flow f is a real number F such that,

$$|f| = F = \sum_{e \in E^+(s)} f(e) - \sum_{e \in E^-(s)} f(e)$$

1.2 Maximum Flow Problem

Given a capacitated graph G(V, E, ρ, ψ), a source vertex s and a sink vertex t, the goal of the problem is to find a s-t flow such that the value of the flow is maximised among all possible s-t flows.

1.2.1 Simple Linear Program definition

This problem can be posed as a linear programming problem. The advantages of this are more than just being able to solve LP's efficiently. The dual of this LP has a very interesting interpretation in graph theory.

$$\max \sum_{u:(s,u)\in E} f((s,u))$$

subject to

- $\forall v \in V \setminus \{s,t\} \sum_{(u,v)\in E} f((u,v)) = \sum_{(v,w)\in E} f((v,w))$
- $\forall (u,v) \in E \; f((u,v)) \leq c((u,v))$
- $\forall (u,v) \in E \; f((u,v)) \geq 0$

Note that a tuple (a, b) represents an edge e whose end points are a and b.

Clearly, the vector $f \in \mathbb{R}^{|E|}$ forms the set of variables in this LP. The constraints are given to ensure that the set of solutions are the set of valid s-t flows. hence, a constraint on the flow conservation on f and the capacity maintanence on f. Hence, for the given problem instance, the number of variables are polynomial in $|E|$ and

$|V|$ and the number of constraints are also polynomial in $|E|$ and $|V|$.

Sometimes, an alternative formulation of the LP is given for this problem.

$$\max \sum_{i \in P} f(i) - \lambda(\sum_{v \in V} F(v))$$

subject to:

- $\forall (j,k) \in E \ \gamma((j,k)) \leq c((j,k))$
- $\forall i \in P$,
$$f(i) \leq F((j,k)) \quad \forall (j,k) \in i$$

CHAPTER 2

A SAMPLE APPENDIX

Just put in text as you would into any chapter with sections and whatnot. Thats the end of it.

Publications

1. S. M. Narayanamurthy and B. Ravindran (2007). Efficiently Exploiting Symmetries in Real Time Dynamic Programming. *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2556–2561.

REFERENCES

Amarel, S., On representations of problems of reasoning about actions. *In* **D. Michie** (ed.), *Machine Intelligence 3*, volume 3. Elsevier/North-Holland, Amsterdam, London, New York, 1968, 131–171.

Barto, A. G., **S. J. Bradtke**, and **S. P. Singh** (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, **72**, 81–138.

Bellman, R. E., *Dynamic Programming*. Princeton University Press, 1957.

Crawford, J. (1992). A theoretical analysis of reasoning by symmetry in first-order logic. URL `citeseer.ist.psu.edu/crawford92theoretical.html`.

Griffiths, D. F. and **D. J. Higham**, *Learning LaTeX*. SIAM, 1997.

Knoblock, C. A., Learning abstraction hierarchies for problem solving. *In* **T. Dietterich** and **W. Swartout** (eds.), *Proceedings of the Eighth National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, California, 1990. URL `citeseer.ist.psu.edu/knoblock90learning.html`.

Kopka, H. and **P. W. Daly**, *Guide to LaTeX (4th Edition)*. Addison-Wesley Professional, 2003.

Lamport, L., *LaTeX: A Document Preparation System (2nd Edition)*. Addison-Wesley Professional, 1994.

Manning, J. B. (1990). *Geometric symmetry in graphs*. Ph.D. thesis, Purdue University.

Ravindran, B. and **A. G. Barto** (2001). Symmetries and model minimization of markov decision processes. Technical report, University of Massachusetts, Amherst.