Lecture Notes in Computer Science 1432
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Stefan Arnborg Lars Ivansson (Eds.)

Algorithm Theory – SWAT'98

6th Scandinavian Workshop
on Algorithm Theory
Stockholm, Sweden, July 8-10, 1998
Proceedings

Springer

Volume Editors

Stefan Arnborg
Lars Ivansson
Nada, KTH
SE-100 44 Stockholm, Sweden
E-mail: {stefan, ivan}@nada.kth.se

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Algorithm theory : proceedings / SWAT '98, 6th Scandinavian Workshop on Algorithm Theory, Stockholm, Sweden, July 8 - 10, 1998. Stefan Arnborg ; Lars Ivansson (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Santa Clara ; Singapore ; Tokyo : Springer, 1998
(Lecture notes in computer science ; Vol. 1432)
ISBN 3-540-64682-5

CR Subject Classification (1991): F.2, G.2, E.1-2, F.1, G.3, I.3.5

ISSN 0302-9743
ISBN 3-540-64682-5 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10637621 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

The papers in this volume were presented at SWAT '98, the Sixth Scandinavian Workshop on Algorithm Theory. The Workshop continues the tradition of previous SWAT Workshops, and the Workshops on Algorithms and Data Structures, WADS, with which it alternates. The Call for Papers sought contributions presenting original research on algorithms and data structures in all areas, including computational geometry, parallel and distributed computing, graph theory, computational biology, and combinatorics. The program committee chose 28 of the 56 submitted papers for presentation. In addition, invited lectures were presented by Andrew V. Goldberg (NEC Research Institute), Johan Håstad (KTH Stockholm), and David Zuckerman (University of Texas at Austin).

SWAT '98 was held in Stockholm, July 8-10, and was organized in cooperation with Kungliga Tekniska Högskolan (the Royal Institute of Technology) and its department of Numerical Analysis and Computing Science, Nada. The organizing committe consisted of Jens Lagergren (KTH Stockholm) and Lars Ivansson (KTH Stockholm).

The program committee wishes to thank all referees who aided us in evaluating the papers.

Finally, we are very grateful to the Swedish Natural Science Research Council and The Swedish Research Council for Engineering Sciences for sponsoring the workshop, and Janne Carlsson the President of the Royal Institute of Technology.

Stockholm, April 1998 Stefan Arnborg

Program Committee

Referees

N. Amenta
G. Andersson
K. Apsitis
L. Arvestad
J. Basch
J. Chen
S. Choi
H. I. Christensen
K. Clarkson
J. Eckerle
S. Edelkamp
L. Engebretsen
K. Eriksson
M. Forsell
P. Fränti
D. Geidmanis
J. Gudmundsson
S. Guha
M. Hammar
S. Hanke
A. Heinz
T. Husfeldt
L. Ivansson
J. Jansson
J. Kaneps
V. Kann
B. M. Kapron
V. Kasturi
J. Katajainen
M. Kiwi
D. Krznaric
J. Larsson
V. Leppänen
V. Liberatore
A. Lingas
B. Lisper
G. Narasimhan
B. Nilsson
R. Rajaraman
S. Ravi Kumar
S. Schuierer
M. Smid
D. Taimina
J. A. Telle
S. Ulfberg
A. Östlin

Table of Contents

Recent Developments in Maximum Flow Algorithms (Invited Lecture)

Andrew V. Goldberg

NEC Research Institute, Inc.
4 Independence Way, Princeton, NJ 08540
avg@research.nj.nec.com

1 Introduction

The maximum flow problem is a classical optimization problem with many applications; see *e.g.* [1,18,39]. Algorithms for this problem have been studied for over four decades. Recently, significant improvements have been made in theoretical performance of maximum flow algorithms. In this survey we put these results in perspective and provide pointers to the literature. We assume that the reader is familiar with basic flow algorithms, including Dinitz' blocking flow algorithm [13].

2 Preliminaries

The maximum flow problem is to find a flow of the maximum value given a graph G with arc capacities, a source s, and a sink t, Here a flow is a function on arcs that satisfies *capacity constraints* for all arcs and *conservation constraints* for all vertices except the source and the sink. For more details, see [1,18,39].

We distinguish between *directed* and *undirected* flow problems. In the standard problem formulation, capacities are reals. Two important special cases of the problem are the integral and the unit capacity cases.

When stating complexity bounds, we denote the number of vertices of the input graph by n and the number of arcs by m. We use the O^* notation to ignore the factors of $\log n$. For the integral case, we assume that capacities are in the interval $[1 \ldots U]$, make the *similarity assumption* [19] $\log U = O(\log n)$, and use the O^{**} notation to ignore the factors of $\log n$ and $\log U$.

The directed and undirected flow problems are closely related. A well-known reduction of Ford and Fulkerson [18] reduces the undirected problem to the directed problem with comparable size and capacity values. A more surprising reduction in the other direction, due to Picard and Ratliff [45], does not significantly increase the problem size but may increase capacities. When applied to a unit capacity capacity problem, the reduction produces a problem with capacities which are polynomial in n (equivalently, an undirected multigraph with unit capacities and the number of edges polynomial in n and m). No efficient

S. Arnborg, L. Ivansson (Eds.): Algorithm Theory - SWAT'98, LNCS 1432, pp. 1–10, 1998.

reduction is known from the undirected unit capacity problem to the directed unit capacity problem.

The *residual capacity* of an arc tells how much the flow on the arc can be increased without violating capacity constraints. A *residual arc* is an arc with a positive residual capacity. Residual arcs induce the *residual graph*. An *augmenting path* is a path from the source to the sink in the residual graph. The *value* of a flow is the total flow into the sink. The *residual flow value* is the difference between the maximum and the current flow values.

3 History

The first maximum flow algorithms, the network simplex method [11] and the augmenting path method [17], were developed in the 50's. These algorithms are pseudopolynomial. In the early 70's, Dinitz [13] and Edmonds and Karp [15] showed that the shortest augmenting path algorithm is polynomial. The main idea of this algorithm is to assign unit lengths to residual arcs and augment flow along a shortest augmenting path. One can show that the source to sink distance is nondecreasing and must increase after polynomially many augmentations.

During the next decade, maximum flow algorithms became faster due to the discovery of powerful new techniques. Dinitz [13] developed the blocking flow method in the context of the shortest augmenting path algorithm; Karzanov [36] stated blocking flows as a separate problem and suggested the use of preflows to solve it. Edmonds and Karp [15] and independently Dinitz [14] developed capacity scaling. Dinitz [14] also shows how to combine blocking flows and capacity scaling to obtain an $O^{**}(nm)$ maximum flow algorithm. (See also [19].) Galil and Naamad [20] developed data structures to speed up flow computations and obtained an $O^{*}(nm)$ algorithm. We use E instead of O when stating expected time bounds.

For the unit capacity case, Karzanov [35] and independently Even and Tarjan [16] show that Dinitz' blocking flow algorithm solves the maximum flow problem in $O(m^{3/2})$ time on multigraphs and in $O(\min(n^{2/3}, m^{1/2})m)$ time on simple graphs.

Table 1 shows the state of the maximum flow time bounds in 1980. Note the gap between the unit capacity and the other cases.

The push-relabel method, first used in [22] and fully developed by Goldberg and Tarjan [26], leads to better theoretical understanding of the problem and substantial improvements in computational performance of the maximum flow codes (see e.g. [5,10,12,23,42]). However, for a long time all bound improvements were limited to logarithmic factors.

Recently, polynomial improvements have been made for two classes of problems: undirected unit capacity problems and integer capacity problems. We discuss these cases below, starting with the latter.

capacities	directed	undirected
unit	$O(\min(n^{2/3}, m^{1/2})m)$ Karzanov [35] Even & Tarjan [16]	$\leftarrow$
integral	$O^{**}(nm)$ Dinitz [14] Gabow [19]	$\leftrightarrow$
real	$O^{*}(nm)$ Galil & Naamad [20]	$\leftrightarrow$

Table 1. The best bounds known in 1980. Arrows show problem reductions: left arrow means the problem reduces to the problem on the left; double arrow means the problem is equivalent to that on the left.

4 Integer Capacity Flows

Table 2 shows the history of time bound improvements for the capacitated maximum flow problem. Dinitz' 1973 algorithm was the first one to achieve an $O^{**}(nm)$ bound. For a long time, the $O(nm)$ bound was the goal of much of the theoretical research in the area. Recently, however, Goldberg and Rao [24] developed an $O^{**}(\min(n^{2/3}, m^{1/2})m)$ algorithm, bridging the gap between the unit and integer capacity cases (see Table 1). This result suggests that the $O(nm)$ goal may be too modest.

The Goldberg-Rao algorithm is a generalization of the blocking flow method that uses a non-unit length function. Recall that the main loop of the blocking flow method finds a blocking flow in an acyclic graph. Such a flow can be found in $O(m \log n)$ time [27]. Recall that the original blocking flow method assigns unit lengths to all residual arcs. The use of non-unit length functions for maximum flow computations has been first suggested by Edmonds and Karp [15]. In the context of minimum-cost flows, the idea has been studied in [47,48]. However, prior to the Goldberg-Rao work, non-unit length functions never gave asymptotic time bound improvements.

Goldberg and Rao use an *adaptive binary length function*.[1] The length of a residual arc depends on the estimated value F of the residual flow. Arcs whose residual capacity is small compared to F have zero lengths and arcs whose residual capacity is large have unit lengths. Zero-length arcs, which seem essential for the algorithm, introduce technical complications, resulting from the fact that the graph G' induced by the residual arcs on shortest paths to the sink may not be acyclic. To deal with this complication, the algorithm contracts strongly connected components of G'. The algorithm limits the amount of flow through each contracted node so that it is easy to route the flow inside the nodes. Because the zero-length arcs have large capacity, the limit is large, and the algorithm makes significant progress even if the limit comes into play.

[1] One can use other length functions.

year	discoverer(s)	bound
1951	Dantzig [11]	$O(n^2mU)$
1956	Ford & Fulkerson [17]	$O(nmU)$
1970	Dinitz [13] Edmonds & Karp [15]	$O(nm^2)$
1970	Dinitz [13]	$O(n^2m)$
1972	Edmonds & Karp [15] Dinitz [14]	$O(m^2 \log U)$
1973	Dinitz [14] Gabow [19]	$O(nm \log U)$
1974	Karzanov [36]	$O(n^3)$
1977	Cherkassky [9]	$O(n^2m^{1/2})$
1980	Galil & Naamad [20]	$O(nm \log^2 n)$
1983	Sleator & Tarjan [46]	$O(nm \log n)$
1986	Goldberg & Tarjan [26]	$O(nm \log(n^2/m))$
1987	Ahuja & Orlin [2]	$O(nm + n^2 \log U)$
1987	Ahuja et al. [3]	$O(nm \log(n\sqrt{\log U}/m))$
1989	Cheriyan & Hagerup [7]	$E(nm + n^2 \log^2 n)$
1990	Cheriyan et al. [8]	$O(n^3/\log n)$
1990	Alon [4]	$O(nm + n^{8/3} \log n)$
1992	King et al. [37]	$O(nm + n^{2+\epsilon})$
1993	Phillips & Westbrook [44]	$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$
1994	King et al. [38]	$O(nm \log_{m/(n \log n)} n)$
1997	Goldberg & Rao [24]	$O(\min(n^{2/3}, m^{1/2})m \log(n^2/m) \log U)$

Table 2. History of the capacitated bound improvements. The dates correspond to the first publication of the result; citations are to the most recent versions. Bounds containing U apply to the integral capacity case only.

Here is a brief outline of the algorithm; see [24] for details. We denote $\min(n^{2/3}, m^{1/2})$ by Λ. The threshold between length one and length zero arcs is $\Theta(F/\Lambda)$.

The algorithm maintains a flow and a cut. The residual capacity of this cut gives an upper bound on the residual flow value. The algorithm also maintains a parameter F, which is an upper bound on the residual flow value. When the algorithm finds a cut with residual capacity of at most $F/2$, it sets F to this residual capacity, thereby decreasing F by at least a factor of two. Distances to the sink are monotone while F does not change. At each iteration, dominated by a blocking flow computation on the contracted graph G', either the flow value increases by at least F/Λ or the source to sink distance increases by at least one. One can show that when the distance exceeds $c\Lambda$, for a fixed constant c, F must decrease. This after $O(\Lambda)$ iterations, F must decrease, either because the flow value increased by at least $F/2$ or because the distance between the source and the sink exceeded $c\Lambda$. This leads to the desired time bound.

5 Undirected Unit Capacity Flows

year	discoverer(s)	bound
1951	Dantzig [11]	$O(n^2m)$
1956	Ford & Fulkerson [17]	$O(nm)$
1973	Karzanov [35] Even & Tarjan [16]	$O(\min(n^{2/3}, m^{1/2})m)$
1997	Karger [30]	$E^*(n^{4/3}m^{2/3})$
1997	Goldberg & Rao [25]	$O(n^{3/2}m^{1/2})$
1998	Karger & Levine [32]	$O(n^{7/6}m^{2/3})$ $E^*(n^{20/9})$

Table 3. History of the bound improvements for the undirected unit capacity problem. The dates correspond to the first publication of the result; citations are to the most recent versions.

In the undirected case, the unit capacity flow bound of $O(\min(n^{2/3}, m^{1/2})m)$ can be improved using *sparsification techniques*. Two different classes of sparcification techniques, originally developed for the minimum cut problem, proved useful for the maximum flow problem: sparse certificates and random sampling. The former was developed by Nagamochi and Ibaraki [41,40]. The latter was developed by Karger [28]. (See also [29,33].)

In this section we assume that the input graph is undirected and all edge capacities are one and all distances are for the unit length function on the residual graph. For simplicity we assume that the input graph is simple and state the bounds in terms of the graph size. For the bounds depending on the flow value and the results for multigraphs, see [30,31,32,34].

The $O(\min(n^{2/3}, m^{1/2})m)$ bound on Dinitz' blocking flow algorithm for the unit capacity case is based on the following two theorems.

Theorem 1. [13] *Each iteration of the blocking flow algorithm increases the distance between the source and the sink.*

Theorem 2. [16,35] *If the distance between the source and the sink is d, then the residual flow value is $O(\min(m/d, (n/d)^2))$.*

5.1 Sparse Certificates

Next we discuss how to use sparse certificates to get a better bound. Let $T_i,\ i \geq 1$, be the set of edges of G constructed as follows. T_1 is the set of edges in a maximal spanning forest of G. For $i > 1$, T_i is the set of edges in the maximal spanning forest of G with $T_1 \cup \ldots \cup T_{i-1}$ deleted. Let $E_k = \bigcup_{i=1}^{k} T_i$. Nagamochi and Ibaraki show that if we are interested in connectivity of k or less, we can restrict our attention to the graph induced by E_k.

Theorem 3. [41] *If the minimum cut capacity between v and w in G is at most k, then the minimum cut capacity between v and w in (V, E_k) is the same as in G.*

Nagamochi and Ibaraki also give an efficient and elegant algorithm to find E_k.

Theorem 4. [40] *For $k = O(n)$, E_k can be found in $O(m)$ time.*

For weighted graphs and large k, E_k can be found in $O(m + n \log n)$ time [41].

Since G_f is a directed graph even if G is undirected, one cannot apply Theorem 3 directly. Consider a flow f in G and let E^0 and E^1 be the edges of G with flow value zero and one, respectively. Edges in E^0 are not used by the flow and therefore undirected, so we can sparsify the corresponding subgraph. Let E^0_k be a k-sparse certificate of (V, E^0).

Theorem 5. [25] *If the residual flow value in G is at most k, then the residual flow value in $(V, E^1 \cup E^0_k)$ is the same as in G.*

We know that $|E^0_k| \leq k(n-1)$. If E^1 is small, we can use the above theorem for sparcification during a maximum flow computation. In many cases, E^1 is small. One example is a flow obtained by the shortest path augmentations [16]. Another case is acyclic flows. The following theorem is a stronger version of one in [21].

Theorem 6. [32] *An acyclic flow f in a simple graph uses $O(n\sqrt{|f|})$ edges.*

Karger and Levine [32] use Theorems 5 and 6 to obtain an $O(n^{7/6}m^{2/3})$ algorithm.

The ideas of this section may lead to better computational results. A computational study to determine practicality of these ideas would be interesting.

5.2 Random Sampling

The currently best randomized algorithm, due to Karger and Levine [32], uses the sparse certificates as well as random sampling techniques. This algorithm uses several nontrivial results developed in a sequence of papers [6,34,30,31,32], and the reader should see these papers for details. We do not attempt an accurate description of the Karger-Levine algorithm, but highlight the mail ideas behind the algorithm and give some intuition behind these ideas.

One additional sparcification technique is captured in the following result, which a variant of a result of Benczúr and Karger [6].

Theorem 7. [30] *For any k, one can, in $O^*(m)$ time, find a set of $O(kn \log n)$ edges whose removal partitions G into k-connected components.*

Intuitively, the graph formed by the removed edges is relatively sparse. Karger [30] uses Theorem 7 to find approximate minimum s-t cuts quickly and uses these s-t cuts in a divide-and-conquer algorithm to find maximum flows quickly when the flow value is small.

capacities	directed	undirected
unit	$O(\min(n^{2/3}, m^{1/2})m)$ Karzanov [35] Even & Tarjan [16]	$O(n^{7/6}m^{2/3})$ $E^*(n^{20/9})$ Karger & Levine [32]
integral	$O^{**}(\min(n^{2/3}, m^{1/2})m)$ Goldberg & Rao [24]	$\leftrightarrow$
real	$O^*(nm)$ Galil & Naamad [20]	$\leftrightarrow$

Table 4. The current state of the maximum flow bounds, ignoring logarithmic factors. Arrows show problem reductions.

The next ingredient of the Karger-Levine algorithm is a fast algorithm for finding flows through k-connected graphs. This algorithm is based on random sampling. We partition the edges of the graph into two sets by sampling the edges with probability one half and recursively find maximum flows in the two resulting subgraphs. Then we apply augmentations to convert the union of the two flows into a maximum flow in the original graph. The following theorem can be used to bound the expected number of augmentations.

Theorem 8. [34] *If G is k-connected and edges are sampled with probability p, then with high probability all cuts in the sampled graph are within $(1 \pm \sqrt{8 \ln n/(pk)})$ of their expected values.*

The Karger-Levine randomized algorithm, and especially its subroutines borrowed from the previous papers, are complicated. It would be interesting to simplify this algorithm without losing efficiency. Simplification may lead to improved theoretical bounds or make the algorithm practical.

6 What Next?

Table 4 summarizes the current state of the maximum flow bounds (modulo logarithmic factors). Comparing to Table 1, one can see that significant progress has been made for the integral and the undirected unit capacity cases. The corresponding algorithms use powerful new design and analysis techniques. Better understanding and refinement of these techniques may lead to even faster algorithms. Although logarithmic factor improvements in time bounds are more likely, polynomial improvements may be possible as well, in particular in the undirected unit capacity case.

Table 4 contains several polynomial gaps between "adjacent" cases. One gap is between the integral and the unit capacity cases. A result of Orlin [43] for minimum-cost flows gives some hope that it may be possible to close this gap.

Another gap is between the undirected and the directed unit capacity cases. Extending the undirected case results to the directed case requires sparcification

techniques for directed graphs. This is a very challenging open problem. Similarly, closing the gap between the unit and the integral undirected cases seems to require directed graph sparcification because of the reduction to the integral directed case.

What is a natural limit of the current techniques? Suppose we have ideal sparsification: in $O(m)$ time, we reduce the number of arcs in the graph we work on to $O(n)$. Furthermore suppose we can close the gaps between the directed and undirected and the integral and real capacity cases. Then we would be able to solve the maximum flow problem in $O^*(m+n^{3/2})$ time. This ambitious goal may be feasible: Goldberg and Rao [24] show that random sampling of [6] can be used to find a cut of capacity within $(1+\epsilon)$ factor of the minimum in $O^*(m+n^{3/2})$ time, for any constant $\epsilon > 0$. (A related result, due to Karger [31], is that a flow of value within $(1-\epsilon)$ of the maximum can be found in $O^*(m\sqrt{n}/\epsilon)$ time.)

We conclude by stating the following open question. Can the maximum flow problem in sparse unit capacity networks be solved in $o(n^{3/2})$ time? A polynomial improvement to the $O(n^{3/2})$ bound for this problem would be a real breakthrough.

Acknowledgements

The author is greatful to Bill Cunningham, David Karger, Maurice Queyranne, and Cliff Stein for their help with this survey.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993. 1, 1
2. R. K. Ahuja and J. B. Orlin. A Fast and Simple Algorithm for the Maximum Flow Problem. *Oper. Res.*, 37:748–759, 1989. 4
3. R. K. Ahuja, J. B. Orlin, and R. E. Tarjan. Improved Time Bounds for the Maximum Flow Problem. *SIAM J. Comput.*, 18:939–954, 1989. 4
4. N. Alon. Generating Pseudo-Random Permutations and Maximum Flow Algorithms. *Information Processing Let.*, 35:201–204, 1990. 4
5. R. J. Anderson and J. C. Setubal. Goldberg's Algorithm for the Maximum Flow in Perspective: a Computational Study. In D. S. Johnson and C. C. McGeoch, editors, *Network Flows and Matching: First DIMACS Implementation Challenge*, pages 1–18. AMS, 1993. 2
6. A. A. Benczúr and D. R. Karger. Approximating s-t Minimum Cuts in $\tilde{O}(n^2)$ Time. In *Proc. 28th Annual ACM Symposium on Theory of Computing*, pages 47–56, 1996. 6, 6, 8
7. J. Cheriyan and T. Hagerup. A randomized maximum flow algorithm. *SIAM Journal on Computing*, 24:203–226, 1995. 4
8. J. Cheriyan, T. Hagerup, and K. Mehlhorn. An $o(n^3)$-time Maximum Flow Algorithm. *SIAM J. Comput.*, 25:1144–1170, 1996. 4
9. B. V. Cherkassky. Algorithm for Construction of Maximal Flows in Networks with Complexity of $O(V^2\sqrt{E})$ Operations. *Mathematical Methods of Solution of Economical Problems*, 7:112–125, 1977. (In Russian). 4
10. B. V. Cherkassky and A. V. Goldberg. On Implementing Push-Relabel Method for the Maximum Flow Problem. *Algorithmica*, 19:390–410, 1997. 2

11. G. B. Dantzig. Application of the Simplex Method to a Transportation Problem. In T. C. Koopmans, editor, *Activity Analysis and Production and Allocation*, pages 359–373. Wiley, New York, 1951. 2, 4, 5
12. U. Derigs and W. Meier. Implementing Goldberg's Max-Flow Algorithm — A Computational Investigation. *ZOR — Methods and Models of Operations Research*, 33:383–403, 1989. 2
13. E. A. Dinic. Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. *Soviet Math. Dokl.*, 11:1277–1280, 1970. 1, 2, 2, 4, 4, 5
14. E. A. Dinic. Metod porazryadnogo sokrashcheniya nevyazok i transportnye zadachi. In *Issledovaniya po Diskretnoĭ Matematike*. Nauka, Moskva, 1973. In Russian. Title translation: Excess Scaling and Transportation Problems. 2, 2, 3, 4, 4
15. J. Edmonds and R. M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. Assoc. Comput. Mach.*, 19:248–264, 1972. 2, 2, 3, 4, 4
16. S. Even and R. E. Tarjan. Network Flow and Testing Graph Connectivity. *SIAM J. Comput.*, 4:507–518, 1975. 2, 3, 5, 5, 6, 7
17. L. R. Ford, Jr. and D. R. Fulkerson. Maximal Flow Through a Network. *Canadian Journal of Math.*, 8:399–404, 1956. 2, 4, 5
18. L. R. Ford, Jr. and D. R. Fulkerson. *Flows in Networks.* Princeton Univ. Press, Princeton, NJ, 1962. 1, 1, 1
19. H. N. Gabow. Scaling Algorithms for Network Problems. *J. of Comp. and Sys. Sci.*, 31:148–168, 1985. 1, 2, 3, 4
20. Z. Galil and A. Naamad. An $O(EV \log^2 V)$ Algorithm for the Maximal Flow Problem. *J. Comput. System Sci.*, 21:203–217, 1980. 2, 3, 4, 7
21. Z. Galil and X. Yu. Short Length Version of Menger's Theorem. In *Proc. 27th Annual ACM Symposium on Theory of Computing*, pages 499–508, 1995. 6
22. A. V. Goldberg. A New Max-Flow Algorithm. Technical Report MIT/LCS/TM-291, Laboratory for Computer Science, M.I.T., 1985. 2
23. A. V. Goldberg. *Efficient Graph Algorithms for Sequential and Parallel Computers.* PhD thesis, M.I.T., January 1987. (Also available as Technical Report TR-374, Lab. for Computer Science, M.I.T., 1987). 2
24. A. V. Goldberg and S. Rao. Beyond the Flow Decomposition Barrier. In *Proc. 38th IEEE Annual Symposium on Foundations of Computer Science*, pages 2–11, 1997. 3, 4, 4, 7, 8
25. A. V. Goldberg and S. Rao. Flows in Undirected Unit Capacity Networks. In *Proc. 38th IEEE Annual Symposium on Foundations of Computer Science*, pages 32–35, 1997. 5, 6
26. A. V. Goldberg and R. E. Tarjan. A New Approach to the Maximum Flow Problem. *J. Assoc. Comput. Mach.*, 35:921–940, 1988. 2, 4
27. A. V. Goldberg and R. E. Tarjan. Finding Minimum-Cost Circulations by Successive Approximation. *Math. of Oper. Res.*, 15:430–466, 1990. 3
28. D. R. Karger. Global Min-Cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm. In *Proc. 4th ACM-SIAM Symposium on Discrete Algorithms*, 1993. 5
29. D. R. Karger. Minimum Cuts in Near-Linear Time. In *Proc. 28th Annual ACM Symposium on Theory of Computing*, pages 56–63, 1996. 5
30. D. R. Karger. Using Random Sampling to Find Maximum Flows in Uncapacitated Undirected Graphs. In *Proc. 29th Annual ACM Symposium on Theory of Computing*, pages 240–249, 1997. 5, 5, 6, 6, 6

31. D. R. Karger. Better Random Sampling Algorithms for Flows in Undirected Graphs. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 490–499, 1998. 5, 6, 8
32. D. R. Karger and M. Levine. Finding Maximum Flows in Undirected Graphs Seems Easier than Bipratire Matching. In *Proc. 30th Annual ACM Symposium on Theory of Computing*, 1997. 5, 5, 6, 6, 6, 6, 7
33. D. R. Karger and C. Stein. A New Approach to the Minimum Cut Problem. *J. Assoc. Comput. Mach.*, 43, 1996. 5
34. D.R. Karger. Random Sampling in Cut, Flow, and Network Design Problems. In *Proc. 26th Annual ACM Symposium on Theory of Computing*, pages 648–657, 1994. Submitted to *Math. of Oper. Res.* 5, 6, 7
35. A. V. Karzanov. O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh. In *Matematicheskie Voprosy Upravleniya Proizvodstvom*, volume 5. Moscow State University Press, Moscow, 1973. In Russian; title translation: On Finding Maximum Flows in Networks with Special Structure and Some Applications. 2, 3, 5, 5, 7
36. A. V. Karzanov. Determining the Maximal Flow in a Network by the Method of Preflows. *Soviet Math. Dok.*, 15:434–437, 1974. 2, 4
37. V. King, S. Rao, and R. Tarjan. A Faster Deterministic Maximum Flow Algorithm. In *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 157–164, 1992. 4
38. V. King, S. Rao, and R. Tarjan. A Faster Deterministic Maximum Flow Algorithm. *J. Algorithms*, 17:447–474, 1994. 4
39. E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Reinhart, and Winston, New York, NY., 1976. 1, 1
40. H. Nagamochi and T. Ibaraki. A Linear-Time Algorithm for Finding a Sparse k-Connected Spanning Subgraph of a k-Connected Graph. *Algorithmica*, 7:583–596, 1992. 5, 6
41. H. Nagamochi and T. Ibaraki. Computing Edge-Connectivity in Multigraphs and Capacitated Graphs. *SIAM J. Disc. Math.*, 5:54–66, 1992. 5, 6, 6
42. Q. C. Nguyen and V. Venkateswaran. Implementations of Goldberg-Tarjan Maximum Flow Algorithm. In D. S. Johnson and C. C. McGeoch, editors, *Network Flows and Matching: First DIMACS Implementation Challenge*, pages 19–42. AMS, 1993. 2
43. J. B. Orlin. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. *Oper. Res.*, 41:338–350, 1993. 7
44. S. Phillips and J. Westbrook. Online Load Balancing and Network Flow. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, pages 402–411, 1993. 4
45. J. C. Picard and H. D. Ratliff. Minimum Cuts and Related Problems. *Networks*, 5:357–370, 1975. 1
46. D. D. Sleator and R. E. Tarjan. A Data Structure for Dynamic Trees. *J. Comput. System Sci.*, 26:362–391, 1983. 4
47. C. Wallacher. A Generalization of the Minimum-Mean Cycle Selection Rule in Cycle Canceling Algorithms. Technical report, Preprints in Optimization, Institute für Angewandte Mathematik, Technische Universität Braunschweig, Germany, 1991. 3
48. C. Wallacher and U. Zimmermann. A Combinatorial Interior Point Method for Network Flow Problems. Technical report, Preprints in Optimization, Institute für Angewandte Mathematik, Technische Universität Braunschweig, Germany, 1991. 3

An ϵ - Approximation Algorithm for Weighted Shortest Paths on Polyhedral Surfaces *

Lyudmil Aleksandrov[1], Mark Lanthier[2], Anil Maheshwari[2], Jörg-R. Sack[2]

[1] Bulgarian Academy of Sciences, CICT, Acad. G. Bonchev Str. Bl. 25-A, 1113 Sofia, Bulgaria
[2] School of Computer Science, Carleton University, Ottawa, Ontario K1S5B6, Canada

Abstract. Let $\mathcal{P}$ be a simple polyhedron, possibly non-convex, whose boundary is composed of n triangular faces, and in which each face has an associated positive weight. The cost of travel through each face is the distance traveled multiplied by the face's weight. We present an ϵ-approximation algorithm for computing a weighted shortest path on $\mathcal{P}$, i.e. the ratio of the length of the computed path with respect to the length of an optimal path is bounded by $(1+\epsilon)$, for a given $\epsilon > 0$.We give a detailed analysis to determine the exact constants for the approximation factor. The running time of the algorithm is $O(mn \log mn + nm^2)$. The total number of Steiner points, m, added to obtain the approximation depends on various parameters of the given polyhedron such as the length of the longest edge, the minimum angle between any two adjacent edges of $\mathcal{P}$ and the minimum distance from any vertex to the boundary of the union of its incident faces and the ratio of the largest (finite) to the smallest face weights of $\mathcal{P}$. Lastly, we present an approximation algorithm with an improved running time of $O(mn \log mn)$, at the cost of trading off the constants in the path accuracy. Our results present an improvement in the dependency on the number of faces, n, to the recent results of Mata and Mitchell [10] by a multiplicative factor of $n^2/\log n$, and to that of Mitchell and Papadimitriou [11] by a factor of n^7.

1 Introduction

1.1 Problem Definition

Shortest path problems are among the fundamental problems studied in computational geometry and other areas such as graph algorithms, geographical information systems (GIS) and robotics.[5] Let s and t be two vertices on a given possibly non-convex polyhedron $\mathcal{P}$, in $\Re^3$, consisting of n triangular faces on its

* Research supported in part by ALMERCO Inc. & NSERC

[5] We encountered several shortest path related problems in our R&D on GIS (see [15]); more specifically, e.g., in emergency response time modeling where emergency units are dispatched to emergency sites based on minimum travel times.

S. Arnborg, L. Ivansson (Eds.): Algorithm Theory - SWAT'98, LNCS 1432, pp. 11–22, 1998.

boundary, each face has an associated weight, denoted by a positive real number w_i. A Euclidean *shortest path* $\pi(s,t)$ between s and t is defined to be a path with minimum Euclidean length among all possible paths joining s and t that lie on the surface of $\mathcal{P}$. A *weighted shortest path* $\Pi(s,t)$ between s and t is defined to be a path with minimum cost among all possible paths joining s and t that lie on the surface of $\mathcal{P}$. The *cost* of the path is the sum of lengths of all segments, the path traverses in each face multiplied by the corresponding face weight. A path $\Pi'(s,t)$ between two points s and t is said to be an ϵ-approximation of a (true) shortest path $\Pi(s,t)$ between s and t, if $\frac{\Pi'(s,t)}{\Pi(s,t)} \leq 1+\epsilon$, for some $\epsilon > 0$. The problem addressed in this paper is to determine an ϵ-approximate shortest path between two vertices on a weighted polyhedron.

1.2 Related Work

Shortest path problems in computational geometry can be categorized by various factors which include the dimensionality of space, the type and number of objects or obstacles (e.g., polygonal obstacles, convex or non-convex polyhedra, ...), and the distance measure used (e.g., Euclidean, number of links, or weighted distances). Several research articles, including surveys (see [5,13]), have been written presenting the state-of-the-art in this active field. Due to the lack of space, here we discuss those contributions which relate more directly to our work; these are in particular 3-dimensional weighted scenarios.

Mitchell and Papadimitriou [11] introduced the weighted region problem in which each face has an associated positive weight, denoted by a real number $w_i > 0$. They presented an algorithm that computes a path between two points in a weighted planar subdivision which is at most $(1+\epsilon)$ times the shortest weighted path cost. Their algorithm requires $O(n^8 L)$ time in the worst case, where $L = \log(nNW/w\epsilon)$ is a factor representing the bit complexity of the problem instance. Here N is the largest integer coordinate of any vertex of the triangulation and W (w) is the maximum (minimum) weight of any face of the triangulation. Johannson discussed a weighted distance model for injection molding [6]. Lanthier et al. [8] presented several practical algorithms for approximating shortest paths in weighted domains. In addition, to their experimental verification and time analysis, they provided theoretically derived bounds on the quality of approximation. More specifically, the cost of the approximation is no more than the shortest path cost plus an (additive) factor of $W|L|$, where L is the longest edge, W is the largest weight among all faces. They also used graph spanners to get at most β times the shortest path cost plus $\beta W|L|$, where $\beta > 1$ is an adjustable constant. Mata and Mitchell [10] presented an algorithm that constructs a graph (pathnet) which can be searched to obtain an approximate path; their path accuracy is $(1+\frac{W}{kw\theta_{min}})$, where θ_{min} is the minimum angle of any face of $\mathcal{P}$, W/w is the largest to smallest weight ratio and k is a constant that depends upon ϵ.

Table 1 compares the running times for the ϵ-approximation algorithms developed by [11], [10], and the one presented in this paper in the case where all