Design and Analysis of Efficient Algorithms to Solve the Maximum Concurrent Flow Problem

Farhad Shahrokhi
Western Michigan University

Follow this and additional works at: https://scholarworks.wmich.edu/dissertations
Part of the Mathematics Commons

Recommended Citation
Shahrokhi, Farhad, "Design and Analysis of Efficient Algorithms to Solve the Maximum Concurrent Flow Problem" (1986).
Dissertations. 2262.
https://scholarworks.wmich.edu/dissertations/2262
DESIGN AND ANALYSIS OF EFFICIENT ALGORITHMS TO SOLVE
THE MAXIMUM CONCURRENT FLOW PROBLEM

by

Farhad Shahrokhi

A Dissertation
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirement for the
Degree of Doctor of Philosophy
Department of Mathematics and Statistics

Western Michigan University
Kalamazoo, Michigan
December 1986
DESIGN AND ANALYSIS OF EFFICIENT ALGORITHMS TO SOLVE
THE MAXIMUM CONCURRENT FLOW PROBLEM

Farhad Shahrokhi, Ph.D.
Western Michigan University, December 1986

The maximum concurrent flow (MCFP) is a generalized commodity flow problem, where every pair of entities can send and receive flow [Ma85], [BM86], [MS86]. We develop efficient labeling algorithms to solve the MCFP. We explore the combinatorial structure of the MCFP and show that the problem of associating costs (distances) to the edges so as to maximize the minimum cost of routing the concurrent flow is the dual of the MCFP. This duality covers max-flow min-cut theorem as a special case. Applications in packet switched networks [At81] and cluster analysis [Ma86] are discussed.
INFORMATION TO USERS

While the most advanced technology has been used to photograph and reproduce this manuscript, the quality of the reproduction is heavily dependent upon the quality of the material submitted. For example:

- Manuscript pages may have indistinct print. In such cases, the best available copy has been filmed.

- Manuscripts may not always be complete. In such cases, a note will indicate that it is not possible to obtain missing pages.

- Copyrighted material may have been removed from the manuscript. In such cases, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, and charts) are photographed by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each oversize page is also filmed as one exposure and is available, for an additional charge, as a standard 35mm slide or as a 17”x 23” black and white photographic print.

Most photographs reproduce acceptably on positive microfilm or microfiche but lack the clarity on xerographic copies made from the microfilm. For an additional charge, 35mm slides of 6”x 9” black and white photographic prints are available for any photographs or illustrations that cannot be reproduced satisfactorily by xerography.
Shahrokhi, Farhad

DESIGN AND ANALYSIS OF EFFICIENT ALGORITHMS TO SOLVE THE MAXIMUM CONCURRENT FLOW PROBLEM

Western Michigan University

University Microfilms International

300 N. Zeeb Road, Ann Arbor, MI 48106

Ph.D. 1987
PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark ✓.

1. Glossy photographs or pages
2. Colored illustrations, paper or print
3. Photographs with dark background
4. Illustrations are poor copy
5. Pages with black marks, not original copy
6. Print shows through as there is text on both sides of page
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements
9. Tightly bound copy with print lost in spine
10. Computer printout pages with indistinct print
11. Page(s) ___________ lacking when material received, and not available from school or author.
12. Page(s) ___________ seem to be missing in numbering only as text follows.
13. Two pages numbered. Text follows.
14. Curling and wrinkled pages
15. Dissertation contains pages with print at a slant, filmed as received
16. Other

University Microfilms International

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Professor David W. Matula, for guiding me and sharing his valuable ideas with me. Special thanks are also due to Professor John Petro, Professor Gary Chartrand, Professor Dalia Motskin, and, finally, Professor Richard Helgason for serving on my committee.

Farhad Shahrokhi
# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** ........................................................................................................... ii

**LIST OF FIGURES** ...................................................................................................................... iv

**CHAPTER**

I. INTRODUCTION ................................................................................................................................. 1

II. THE MAXIMUM CONCURRENT FLOW PROBLEM (MCFP) .......................................................... 3

   2.1 Problem Definition ....................................................................................................................... 3

   2.2 The Structure of the MCFP ......................................................................................................... 7

III. THE MCFP WITH UNIFORM CAPACITY ..................................................................................... 12

   3.1 A Flow Routing Algorithm ....................................................................................................... 13

   3.2 Data Structure ............................................................................................................................ 15

   3.3 Performance R ............................................................................................................................ 17

   3.4 Empirical Time and Storage Complexity .................................................................................... 18

   3.5 Applications of the MCFP to the SCP ....................................................................................... 19

IV. AN $e$ - APPROXIMATION ALGORITHM ...................................................................................... 21

V. THE MCFP WITH INIFORM DEMAND .......................................................................................... 30

VI. THE MCFP and THE SINGLE COMMODITY FLOW PROBLEM .................................................. 37

   6.1 The Single Commodity Flow Problem ..................................................................................... 37

   6.2 Duality of the MCFP ................................................................................................................... 39

VII. APPLICATIONS OF THE MCFP ................................................................................................. 43

   7.1 The Routing Problem in Telecommunications ........................................................................... 43

   7.2 Applications of the MCFP to the Routing Problem ....... 44

   7.3 Applications of Duality Theorem to the Routing Problem ...................................................... 45

   7.4 Applications of the MCFP to Cluster Analysis ........................................................................... 46

**APPENDIX** .................................................................................................................................... 50

**BIBLIOGRAPHY** ............................................................................................................................. 57
LIST OF FIGURES

1. Data Structure for Algorithm R ...................................... 16
CHAPTER I

INTRODUCTION

Consider a network of entities (cities, computers, etc.) in which there exists a demand for flow between all pairs of entities. The flow is sustained through channels with certain capacities. It is desired to assign flow to each path (route) of the network, such that the ratio of the flow supplied between every pair of entities to the demand for that pair (throughput) is the same for all pairs of entities. This flow assignment must respect the capacity constraints, that is, the flow on a channel should not exceed its capacity. The maximum concurrent flow problem (MCFP) is to assign flow to the routes such that the throughput is maximized. For a graph model of this problem the vertices represent entities, the edges correspond to the channels, and paths represent the flow routes. The capacities are represented by the weights assigned to the edges.

Chapter 2 formally defines the MCFP. The linear programming (LP) formulations of the MCFP are presented and their deficiencies are explained. The MCFP is shown to be solvable by a polynomially time bounded algorithm that is not efficient. Certain properties of the MCFP that are useful for the design of an efficient algorithm to solve the problem are explored.

In Chapter 3 we present a routing algorithm to solve the instances of the MCFP with a constant capacity function (the MCFP with uniform capacity). Empirical results are presented and are compared to the results obtained from LP and other current techniques from the literature. Our algorithm is shown to be superior to other methods for solving the MCFP. Empirical time and space complexities are presented.

Chapter 4 is considered to be the most important part of this work. We present an approximation algorithm to solve the MCFP with uniform demand and prove its correctness.
The algorithm is shown to approximate the solution to the MCFP within any (non-zero) arbitrary small tolerance. Results regarding the time complexity are given.

In chapter 5 a useful transformation between an instance of the MCFP with uniform demand, where all the demands are the same, and an instance of the MCFP with uniform capacity is developed. This transformation enables us to solve efficiently the instances of the MCFP with uniform demand.

In Chapter 6 we show the single commodity flow problem is a special case of the MCFP. We give a new proof of the max-flow min-cut theorem, using the structure of the MCFP. A duality theorem for the MCFP is presented and shown to cover the max-flow min-cut theorem as a special case. The proof for our duality theorem is constructive and uses the algorithm presented in Chapter 4.

Finally, chapter 7 is devoted to applications of the MCFP. A survey of the routing algorithms in packet switched networks is presented. The results are compared to our routing algorithm (described in Chapter 3). The conclusion is that our routing algorithm is a promising routing policy for the packet switched networks. Also, in this chapter, applications of the MCFP in cluster analysis are discussed and empirical results are presented. Our empirical results indicate that the MCFP can solve many instances of the top-down clustering problem which is NP-hard.
CHAPTER II

THE MAXIMUM CONCURRENT FLOW PROBLEM (MCFP)

Section 2.1 Problem Definition

Let $G = (V, E)$ be a graph. We denote by $P_e$ the set of all paths containing an edge $e$, for all edges $e \in E$, and by $P_{ij}$ the set of all paths between end vertices $i,j$, for all distinct pair of vertices $i, j \in V$. Also, let $P$ denote the set of all nontrivial paths in $G$. A capacity function on $G$ is a function $C : E \rightarrow \mathbb{R}^+$. A demand function $D$ on $G$ is a function $D : V \times V \rightarrow \mathbb{R}^+$ such that $D(i, j) = D(j, i)$ and $D(i, i) = 0$ for all $i, j \in V$.

A concurrent flow of throughput $Z$ in $G$ is a function $f : P \rightarrow \mathbb{R}^+$ such that

1. $\sum_{p \in P_{ij}} f(p) = ZD(i, j)$ for all distinct pairs of vertices $i, j \in V$ and
2. $\sum_{p \in P_e} f(p) \leq C(e)$ for every edge $e \in E$,

where $D$ and $C$ are the given demand and capacity functions on $G$. Consider the graph $K_{1,3}$. The flow assignment of $1/3$ to the unique paths between each pair of vertices will yield a throughput of $1/3$.

The maximum concurrent flow problem (MCFP) is to find the largest throughput $Z$ achieved by a concurrent flow function (the maximum concurrent flow) $f$, which is explicitly defined on $P'$ (or the set of active paths), where $P' \subseteq P$ such that $f(p) > 0$ if $p \in P'$. A path is called active if it belongs to $P'$. The set of saturated edges for $f$ is $\{e|f(e) = C(e)\}$, where given any concurrent flow $f$, $f(e)$ denotes $\sum_{p \in P_e} f(p)$ for all $e \in E$.

Given $G, D$ and $C$, the set of critical edges in $G$ consists precisely of all those edges that are saturated under any maximum concurrent flow function in $G$.

Theorem 2.1. The MCFP is solvable in polynomially bounded time and space. Furthermore, the number of the active paths in the solution is polynomially bounded by the length of input.
Proof. The MCFP can be expressed as the linear program,

\[
\begin{align*}
\text{Maximize } & Z, \\
\text{subject to } & \sum_{p \in P_{i,j}} f(p) - ZD(i,j) = 0 \quad \text{for all distinct } i, j \in V, \\
& \sum_{p \in P_{i,j}} f(p) \leq C(e) \quad \text{for all } e \in E,
\end{align*}
\]

(2.1)

where \(D, C\) are the demand and capacity functions on \(G = (V, E)\). This formulation may use \(O(|V|^3)\) variables and therefore cannot be used directly for a polynomially time bounded solution of the MCFP. We shall now develop a multicommodity flow model and show it is equivalent to the MCFP. Furthermore, we show that this multicommodity flow problem is solvable in polynomial time and prove that its solution can be transformed to a solution of the MCFP in polynomially bounded time.

Replace each edge of \(G\) by a pair of arcs in opposite directions to obtain a digraph \(G' = (V, D')\). Denote by \(N^+(x)\) (respectively, \(N^-(x)\)) the set of vertices of \(D\) adjacent from (respectively adjacent to) \(x\), for all \(x \in V\). Now consider the following flow problem:

\[
\begin{align*}
\text{Maximize } & Z, \\
\text{subject to } & \sum_{i < j} f^{ij}_{ik} - \sum_{i < j} f^{ji}_{ik} = \begin{cases} 
ZD(i,j) & \text{if } i = i; \\
-ZD(i,j) & \text{if } i = j \text{ for } 1 \leq i \leq n, 1 \leq j \leq n, \\
0 & \text{otherwise},
\end{cases} \\
& \sum_{i < j} f^{ij}_{ik} + f^{ji}_{ik} \leq C(e) \quad \text{for all edges } (e = ik) \text{ in } E,
\end{align*}
\]

(2.2)

where \(f^{ij}_{ik}\) is the flow of the commodity \(ij\) on the arc \(kl\) of \(D\), for all \(i < j, i, j \in V\). Considering this new problem as \(\binom{|V|}{2}\) single commodity problems and utilizing the standard procedure from [FF62] to decompose the flow on the directed paths for each commodity, it follows that an optimal solution of either (2.1) or (2.2) yields an optimal solution to the other with the same throughput. We refer to (2.1) as the edge path formulation and to (2.2) as the node arc formulation of the MCFP, which is consistent with the terminology used for the single commodity flow problem in [FF62]. The node arc formulation of the MCFP uses \(|V| \binom{|V|}{2} + |E|\) constraints and \(2|E| \binom{|V|}{2} + 1\) variables, thus the storage complexity is \(O(|V|^3 |V|^3)\). The ellipsoidal version of linear programming is a polynomially bounded algorithm [Ps82], indicating the node arc formulation of the MCFP is solvable in polynomial time \(O(|E|^3 |V|^3 Q)\), where \(Q\) is the number of binary digits to represent

\[
\max \{\max_{i,j} D(i,j), \max_{e \in E} C(e)\}.
\]
We now present an algorithm which determines a solution to the MCFP from a solution to the node arc formulation of the MCFP in polynomially bounded time and space.

Algorithm T

To transform an optimal solution for the node arc formulation of the MCFP to a solution of the MCFP. / 

The input is $G'$ and an optimal solution to the node arc formulation of the MCFP. The output is a maximum concurrent flow function $\hat{f}$, defined on $P'$, $P' \subseteq P$ such that $\hat{f}(p) > 0$ for all $p \in P'$.

BEGIN

$P' = \emptyset$.

FOR all commodities $ij$, $i, j \in V$, $i < j$

Construct a digraph $H = \langle V, D \rangle$ such that $a \in D$ iff $f_a^i > 0$.

WHILE there exists a directed ij path $p$ in $H$

Set $\delta = \min\{f_a^i \mid p \text{ contains arc } a\}$.

$\hat{f}(p) = \delta$.

FOR all arcs $a \in p$

$f_a^i = f_a^i - \delta$.

IF $f_a^i = 0$ THEN Delete the arc $a$ from $H$.

ENDFOR

$P' = P' \cup \{p\}$.

ENDWHILE

ENDFOR

END

For each commodity $ij$, the construction of $H$ is done in time $O(|V|^2)$. The WHILE statement at most iterates $|D'| = 2|E|$ times, since at each iteration the flow of the commodity
ij is set to zero on one of the arcs of $H$. The outer FOR loop runs exactly $\binom{|V'|}{2}$ times and the Breadth First Search can be used to find a directed path satisfying the conditions of the WHILE statement (or to confirm that such a path does not exist). Thus we conclude that the algorithm $T$ halts and has the order of time complexity

$$O(|V'|^2 |E|^2).$$

Furthermore, upon termination: $|P'| \leq 2\binom{|V'|}{2} |E|$, therefore the storage requirement is

$$O(|V'|^3 |E|).$$

We have introduced a more efficient algorithm than the above for the conversion of the node arc formulation to the edge path formulation in [MS86] which is not discussed here, since the dominating bounds for the time and space complexities to solve the MCFP are those of the ellipsoidal method to solve node arc form. Now ignore the directions of the arcs in $P'$. Then $P' \subseteq P$ and the results in [FF62] guarantee that $\vec{f}$ is a maximum concurrent flow in $G$ such that $\vec{f}(p) > 0$, for all $p \in P'$. It follows that the MCFP is solved in time

$$O(|E|^3 |V|^6 Q)$$

and space

$$O(|V'|^6 |E|). \quad \textbf{Q.E.D.}$$

The method suggested in Theorem 2.1, although polynomially space and time bounded, applies practically to the problems of very small size. The edge path formulation of the MCFP is solved using linear programming techniques which use special network structures [He83,Pa84] for graphs of at most 40 vertices. Current flow routing algorithms [BM86, TM86] do not solve problems of size 30 vertices or more. Therefore the need to devise algorithms solving large MCFP's is obvious.
The edge path formulation of the MCFP as presented in (2.1) indicates the existence of a basic optimal solution. Since the number of constraints in (2.1) is \( \binom{n}{2} + |E| \), it follows that there exists a solution to the MCFP with only \( \binom{n}{2} + |E| \) active paths. This, however, implies that the average number of paths per pair of vertices in such a solution is at most two. This rather surprising result implies that the linear programming method needs too many paths to trigger a solution; however, only very few of these paths will be active in the final solution.

A challenging task is to find a method of solving the problem such that the number of paths needed to represent the problem and the solution is small (perhaps about an average of two per pair). We will discuss efficient algorithms which only utilize an average of three paths per pair in Chapter 3.

Section 2.2 The Structure of the MCFP

In this section we discuss the structure of the MCFP which will help us to design more efficient algorithms to solve the problem.

**Lemma 2.2.** Any concurrent flow \( f \) of throughput \( Z \) in the graph \( G \) with demand \( D \) and capacity \( C \) satisfies

\[
Z \leq \frac{\sum_{e \in (A, \bar{A})} C(e)}{\sum_{i \in A, j \in \bar{A}} D(i, j)},
\]

where \((A, \bar{A})\) is any cut in \( G \).

**Proof.** The total amount of flow supplied between \( A \) and \( A \) is \( Z \sum_{i \in A, j \in \bar{A}} D(i, j) \), while the total capacity available between \( A \) and \( \bar{A} \) is \( \sum_{e \in (A, \bar{A})} C(e) \); therefore

\[
Z \sum_{i \in A, j \in \bar{A}} D(i, j) \leq \sum_{e \in (A, \bar{A})} C(e)
\]

or

\[
Z \leq \frac{\sum_{e \in (A, \bar{A})} C(e)}{\sum_{i \in A, j \in \bar{A}} D(i, j)}.
\]

Q.E.D.

A distance (toll) function on \( G = < V, E > \) is a function \( d : E \to R^+ \). Let \( d \) be a distance function on \( G \).
We extend the domain of $d$ to $E \cup V \times V \cup P$ by

$$d(p) = \sum_{e \in P} d(e) \text{ for all } p \in P.$$ 

$$d(i, j) = \min \{d(p) | p \text{ is an } ij \text{ path} \} \text{ for all } i, j \in V$$

**Lemma 2.3.** Let $d$, $D$, and $C$ denote the distance, the demand, and the capacity functions on a graph $G$, respectively. Then

$$Z \leq \frac{\sum_{e \in E} d(e) C(e)}{\sum_{i < j} d(i, j) D(i, j)},$$

(2.4)

where $Z$ denotes the throughput of any concurrent flow $f$ in $G$.

**Proof.** Consider the expression $\sum_{p \in P} d(p) f(p)$. Then

$$\sum_{p \in P} d(p) f(p) \geq \sum_{i < j} d(i, j) \sum_{p \in P_{ij}} f(p) = Z \sum_{i < j} d(i, j) D(i, j).$$

Furthermore

$$\sum_{p \in P} d(p) f(p) = \sum_{e \in E} d(e) \sum_{p \in P_e} f(p) \leq \sum_{e \in E} d(e) C(e).$$

It follows that

$$Z \leq \frac{\sum_{e \in E} d(e) C(e)}{\sum_{i < j} d(i, j) D(i, j)}. \text{ Q.E.D.}$$

Inequalities (2.3) and (2.4) provide upper bounds on $Z$ that will be used in our algorithms in chapters 3 and 4. The concept of the distance function will provide a duality theory for the MCFP (see Chapter 5). This duality provides an understanding of the structure of the MCFP based on the interpretation of a dual problem, which is implicitly solved. Our duality theory is analogous to that of the single commodity network flow [FF62] and is motivated by the initial research in [Ma85].

For a graph $G$ with demand function $D$ and capacity function $C$, let

$$C(A, A) = \sum_{e \in (A, A)} C(e),$$

$$D(A, A) = \sum_{i \in A, j \in A} D(i, j) \text{ and}$$

$$\text{den}(A, A) = \frac{C(A, A)}{D(A, A)}. $$
where \((A, \bar{A})\) is any cut in \(G\). Notice that \(\text{den}(A, \bar{A})\) represents the density of the capacity relative to the demand across the cut \((A, \bar{A})\). A sparsest cut is a cut of minimum density. Observe that (2.3) indicates that the optimal throughput value of the MCFP is bounded above by the density of a sparsest cut. It is interesting to indicate that the sparsest cut problem, which is to determine a sparsest cut and its density, is computationally hard, while by Theorem 2.1 the MCFP is solvable in polynomial time. We have shown the following in [MS86].

**Theorem 2.4.** The sparsest (densest) cut problem is NP-hard.

Let \(G\) be a graph. We denote by \((A_1, A_2, \ldots, A_k)\), for some \(k \geq 2\), the set of all edges in \(G\) whose end vertices are in distinct elements of the partition \(\{A_1, A_2, \ldots, A_k\}\) of \(V\). We name \((A_1, A_2, \ldots, A_k)\) a \(k\)-partite cut in \(G\). The following theorem explores the structure of the critical edges.

**Theorem 2.5.** Let \(G, D, C\) be an instance of the MCFP. Denote by \(\hat{f}\) a maximum concurrent flow, by \(E_s\) the set of all saturated edges under \(\hat{f}\), and finally by \(E_c\) the set of critical edges.

Then

(a) \(G - E_s\) is disconnected and

(b) \(E_c\) is a \(k\)-partite cut, for some \(k \geq 2\).

**Proof of (a)** Assume, to the contrary that \(G - E_s\) is connected. Then there is a spanning tree \(T\) of \(G\) such that none of the edges in \(T\) are saturated under \(\hat{f}\). Let \(\epsilon = \text{Min}\{C(e) - \hat{f}(e) | e \in E(T)\}\)

and \(\epsilon_{ij} = \frac{C(i,j)}{M}\) where \(M = \text{Max}\{D(i,j) | i, j \in V\}\). Define \(f: P \rightarrow R^+\) by

\[
\begin{align*}
    f(p) = \begin{cases} 
    \hat{f}(p) & \text{if } p \text{ is not a path of the tree } T, \\
    \hat{f}(p) + \epsilon_{ij} & \text{if } P \text{ is an } i-j \text{ path in the tree } T.
    \end{cases}
\end{align*}
\]

Observe that

\[
f(e) = \sum_{p \in P_e} f(p) = \sum_{p \in P_e} \hat{f}(p) \leq C(e) \quad \text{for all } e \in E - E(T)
\]

and

\[
f(e) = \sum_{p \in P_e} f(p) \leq \sum_{p \in P_e} \hat{f}(p) + \sum_{i,j} \epsilon_{ij} \leq
\]
Therefore

\[ f(e) + \frac{\binom{|V| - 1}{2}}{|V|} \epsilon M = \hat{f}(e) + \epsilon \leq C(e) \text{ for all } e \in E(T) \]

Therefore

\[ f(e) = \sum_{p \in P} f(p) \leq C(e) \text{ for all } e \in E. \]

Now notice that

\[ \sum_{p \in P} f(p) = \sum_{p \in P} (\hat{f}(p) + \epsilon) = 2D(i,j) + \frac{D(i,j)\epsilon}{\binom{|V|}{2}M} = D(i,j) \left( Z + \frac{\epsilon}{\binom{|V|}{2}M} \right). \]

Therefore \( f \) is a concurrent flow in \( G \) with a throughput more than \( \hat{Z} \), which is a contradiction, implying that \( G - E_c \) is disconnected.

Proof of (b) For every \( e \in E - E_c \) there must be a maximum concurrent flow function \( f_e \) such that \( f_e(e) < C(e) \) (i.e., \( f_e \) does not saturate \( e \), since \( e \) is not critical). Define \( f : P \to R^+ \) by

\[ f(p) = \frac{\sum_{e \in E - E_c} f_e(p)}{|E| - |E_c|}. \]

Then \( f \) is a maximum concurrent flow in \( G \) only saturating the edges in \( E_c \). By part (a), \( G - E_c \) is disconnected. Let \( C_1, C_2, \ldots, C_k \) denote the vertex sets of the components of \( G - E_c \). We have \( (C_1, C_2, \ldots, C_k) \subseteq E_c \). We will show that \((C_1, C_2, \ldots, C_k) = E_c \). Assume, to the contrary, that \((C_1, C_2, \ldots, C_k) \subset E_c \). Let \( e = t_l \in E_c - (C_1, C_2, \ldots, C_k) \). Then vertices \( t, l \) are in \( C_j \) for some \( j, 1 \leq j \leq k \). Since \( C_j \) is connected, there is a \( tl \) path \( \beta \) (in \( C_j \)) containing only unsaturated edges. Let \( \epsilon = \min\{|C(e) - f(e)| \mid e \text{ is an edge in the path } \beta \} \). Then \( \epsilon > 0 \). Let \( p \in P_e \) be an active \( mn \) path, \( m, n \in V \). There exists an \( mn \) path \( q \) such that \( q \) does not contain \( e \), and every edge in \( q \) is either in \( \beta \) or \( p \). Reroute \( \min\{\epsilon, f(p)\} \) units of flow from \( p \) to \( q \), maintaining the optimality of the new concurrent flow and contradicting the criticality of \( e \), which proves part (b). Q.E.D.

We denote the problem of determining the set of critical edges by \( MCFP2 \). The \( k \)-partite cut itself is referred to as the \( MCF \)-partition. We have investigated the relationship between \( MCF \)-partition and sparsest cut problem in [MS86] and have shown the following result.
Theorem 2.6. For a given $G$, $C$, and $D$

(a) The $MCFP_2$ is solvable in polynomially bounded time.

(b) If the $MCF$-partition has at most four parts, then the partition can also identify a sparsest cut.

(c) If $G$ is a tree, then both $MCFP$ and the sparsest cut problems are solved in time complexity $O(|V|^2)$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER III

THE MCFP WITH UNIFORM CAPACITY

In this chapter we describe an iterative Algorithm $R$ to solve those instances of the (MCFP), where the capacity function $C$ is a constant function. This algorithm is extended to more general cases in Chapter 4. Some new concepts are introduced first.

For a given $G$, $D$, and $C$, a concurrent flow of capacity utilization $u$ in $G$ is a function $f : P \rightarrow \mathbb{R}^+$ such that $\sum_{p \in P} f(p) = D(i,j)$ and $\sum_{p \in P} f(p) \leq uC(e)$. The minimum capacity utilization problem (MCUP) is to find the smallest capacity utilization $u^*$ achieved by a concurrent flow capacity utilization function $f^+$. The following lemma shows that the MCFP and MCUP are closely related.

Lemma 3.1. Let $G$, $D$, and $C$ be given. Assume $f^+$ is the solution to the MCUP with the capacity utilization $u^*$. Then $\tilde{f}$ defined by

$$\tilde{f}(p) = \frac{f^+(p)}{u^*} \quad \text{for all} \quad p \in P$$

is a maximum concurrent flow of throughput $\tilde{Z}$ such that $\tilde{Z} = \frac{1}{u^*}$.

Proof. Notice that $\tilde{f}$ is a concurrent flow of throughput $\tilde{Z} = \frac{1}{u^*}$ in $G$. Now assume to the contrary that there exists a concurrent flow $\hat{f}$ of throughput $\hat{Z}$ in $G$ such that $\hat{Z} > \tilde{Z}$. Then $f$ defined by

$$f(p) = \frac{\hat{f}(p)}{\hat{Z}} \quad \text{for all} \quad p \in P$$

is a concurrent flow in $G$ of capacity utilization $u$, with $u = \frac{1}{\hat{Z}} < \frac{1}{\tilde{Z}} = u^*$, which is a contradiction. Therefore $\tilde{f}$ is a maximum concurrent flow in $G$.  

12
Section 3.1 A Flow Routing Algorithm

We now present a heuristic Algorithm $R$ to solve the MCFP in cases where the capacity function is a constant function (the MCFP with uniform capacity). Algorithm $R$ first finds a solution to the MCUP and then transforms this solution to a solution of the MCFP, using Lemma 3.1. Algorithms similar to $R$ are discussed in [BM86], [TM86]. These algorithms are slow and do not solve problems beyond 31 vertices, while $R$ is very fast and has been tested on graphs of up to 120 vertices.

The input to $R$ is the adjacency and the demand matrices for $G$, a constant value $c^*$ for the capacities on the edges, and a tolerance value $\epsilon$, $1 \geq \epsilon > 0$. The output consists of a concurrent flow function $f$ which is defined on the set of active paths and a distance function $d$ such that the throughput $Z$ of $f$ is within the given relative tolerance of the optimal throughput $\hat{Z}$.

Algorithm $R$.

R1. [Determine the initial feasible solution.] Assign $D(i, j)$ units of flow to one shortest path $p \in P_{ij}$ between all distinct pairs of vertices $i, j \in V$. Calculate the value of $f(e)$ which is equal to $\sum_{p \in P_{ij}} f(p)$ for all $e \in E$.

R2. [Find the shortest paths and the upper bounds on the throughput.] Calculate the distance for each edge using

$$d(e) = 2^{\alpha(f(e) - f^*)},$$

where $f^* = \max\{f(e) | e \in E\}$ and the value of $\alpha$ is initially selected such that $d(e) \geq .00001$ for all $e \in E$ to avoid arithmetic underflow. Find the shortest paths (one shortest path per pair of vertices) and set the distance upper bound to

$$\frac{\sum_{e \in E} d(e)c^*}{\sum_{i < j} d(i, j)D(i, j)}.$$

Determine a minimum spanning tree $T$ of $G$ using the distances on the edges (this can be done while finding the shortest paths from vertex 1 to other vertices). Let $e^*$ denote the edge...
with highest distance in \( T \). Calculate the cut upper bound given in (2.2) for the cut \((A, \bar{A})\), where \( \{A, \bar{A}\} \) denotes the partition of \( V \) before \( e^* \) is placed in \( T \). Set the upper bound on the throughput to be the smallest of the distance and the cut upper bounds.

**R3. [Reroute the flow and update.]** Proceeding through each pair \( i, j \) of vertices with \( 1 \leq i \leq j \leq |V| \), determine a longest distance path \( p^* \) currently hosting flow and a shortest distance path \( p_* \), where \( p_*, p^* \in \mathcal{P}_{i,j} \). If \( d(p^*) > d(p_*) \), then reroute

\[
\sigma = \frac{1}{2c} \log_2 \frac{d(p_2)}{d(p_1)}
\]

units of flow from \( p^* \) to \( p_* \), where \( p_2, p_1 \) are the disjoint portions of \( p^* \), \( p_* \), respectively. If \( \sigma > f(p^*) \), then reroute all the flow on \( p^* \) to \( p_* \) and remove path \( p^* \) from the list of paths carrying flow between pair \( ij \). Update the flow on paths \( p_*, p^* \) and the corresponding edges in \( p_*, p^* \). Update \( f^* \).

**R4. [Compute the throughput and check for termination.]** Calculate the normalized throughput realized by this solution using

\[
Z = \frac{f^*}{f_*}
\]

where \( f^* = \text{Max}\{f(e) | e \in E\} \). If the relative difference between the upper bound and the throughput \( Z \) is less than the given tolerance \( \varepsilon \), then set \( f(p) = \frac{Z f^*}{f^*} \) for all active paths \( p \) and stop. Otherwise go to step 5.

**R5. [Check the rate of the convergence.]** If the convergence rate is too slow, change \( \alpha \) to \( 2\alpha \) go to step 2.

The relationship between the MCFP and the MCUP as described in Lemma 3.1 is used by the algorithm. In fact, Algorithm \( R \) solves the MCUP by supplying the demand between all pairs (step 1) and assigning extremely larger distances to the edges hosting larger flows (step 2). This will make the paths containing highly utilized edges less attractive for the future flow assignment, since the flow is always sent on the shortest distance paths. Step 3 is the most important part of the algorithm. In this step the algorithm tries to reduce the utilization.
factor by reducing the flow on highly used (utilized) edges and increasing it on the poorly used (utilized) edges. Step 4 takes advantage of Lemma 3.1 and computes a concurrent flow and its throughput using the current utilization factor which is $\frac{C}{\alpha}$. Finally, step 5 makes the distance function more sensitive in order to easily identify the highly utilized paths. The modification strategy for $\alpha$ will be explained thoroughly in Chapter 4. The cut constructed in step 2 is empirically proven to yield sharp cut upper bounds and to accelerate convergence [TM86].

Section 3.2 Data Structure

We represented the graph using its adjacency list. In order to represent an active path $p \in R_{ij}$, a linked list of nodes was dynamically generated. Each node represents an edge and has five fields to store the two end vertices for the edge, a pointer to the next edge in this path, and finally a forward and a backward pointer for linking all the active paths containing this edge. The set of all active $ij$ paths is represented using a linked list of header nodes, where each node has three fields: one field points to an active $ij$ path, the second field stores the value of flow hosted on the active path pointed to, and finally the last field points to the next header node. Entry $ij$ of a $|V| \times |V|$ matrix (path matrix) is used to point to the beginning of the header list for the pair $ij$. Another $|V| \times |V|$ matrix was used to store the flow on the edges. A third $|V| \times |V|$ matrix (path matrix) is used to identify the set of all active paths containing an edge $e = ij$. Entry $ij$ of this matrix points to a node in the header list. This header node, in return, points to a path containing edge $e$, and the forward and backward pointers in the node representing edge $e$ identify the occurrence of the edge $e$ in the other paths. The $ij$ entry of an auxiliary $|V| \times |V|$ matrix points to the last occurrence of the edge $e = ij$ in the doubly-linked structure. This is particularly useful for updating and representing $P_e$, for each $e \in E$. Note that Algorithm R does not need the edge matrix and doubly-linked list which represents the set of active paths containing a certain edge.
However, this structure will be used in the extensions of the algorithm in Chapter 5.

Figure 1. Data Structure for Algorithm R.
Section 3.3 Performance of R

Algorithm R was implemented using the Pascal language on the VAX 11/780 machine with UNIX 4.2 operating system. We ran our implementation on a wide range of data and summarized the results in Tables 1, 2, and 3 in the appendix. Graphs 1 through 8 in the appendix are selected from the current literature [BM86], [TM86]. Since these benchmark data do not go beyond graphs of 20 vertices, we generated our own data up to graphs of 120 vertices. The data in Table 1 contained graphs 1 through 8, bipartite and random graphs, and cubes. The capacity and demand are both unit functions for all cases in Table 1. Sharpest upper bounds were given by the cut bounds in 13 cases, and by the distance bounds in 7 cases denoted by asterisks. Table 2 contains 15 cases. The underlying graphs are complete and the capacity function is the unit function in all cases. The vertices of the graph are the points in the one or two dimensional Euclidean space. The demand between a pair of vertices equals the distance between that particular pair. The coordinates of the vertices were randomly generated. The plots of the points and the cut identified by the algorithm for some of the interesting cases are given in the appendix. Our results in Tables 1 and 2 certainly show a significant improvement in both the size of the problem solved and computing time compared to [BM86], [PA84], [He85], and [TM86] which only solve the problems of size at most 40 vertices. The tolerance value for all the cases in Tables 1 and 2 is 0.002. Table 3 contains the results for the same graphs in Table 1 with three different tolerance values, $\epsilon = 0.002$, $\epsilon = 0.01$, and $\epsilon = 0.1$. The results in Table 3 indicate that the execution time is not highly dependent on $\epsilon$. That is, making $\epsilon$ ten times smaller does not multiply the execution time by 10. This means that the solution to the MCFP within a very small tolerance value can be obtained (using Algorithm R) in a reasonable amount of time.
Section 3.4 Empirical Time and Storage Complexity

Denote by $m$ the number of iterations that $R$ makes to halt, by $P_t$ the set of active paths at iteration $t$, $i \leq m$, and finally by $|p|$ the number of edges in path $p$. The results in Tables 1 and 2 indicate that the average number of paths per pair calculated by

$$Max \{ \frac{|P_t|}{|V|^2} \}$$

is always less than 3. Our findings also show that (see Table 1) the average path length (i.e., average number of edges in a path) computed by $Max \{ \frac{\sum_{p \in P_t} |p|}{|P_t|} \}$, is always less than 4. It follows that at any iteration $t$, $\frac{|P_t|}{|V|^2} \sum_{p \in P_t} |p| = \frac{\sum_{p \in P_t} |p|}{|V|^2} \leq 12$, implying that the storage requirement is $O(|V|^2)$.

It is clear that all the steps in the algorithm except for step 3 can be done in $O(|V|^3)$. We cannot theoretically bound the time complexity for step 3. However the empirical storage requirement is $O(|V|^2)$ and the rerouting (including finding the disjoint parts of the two paths) can always be carried out in $O(|V|)$. Thus we conclude that the order of time complexity for this step is $O(|V|^3)$. The number of iterations to reach convergence within a reasonable tolerance (about 0.002) is always less than $|V|$, therefore the time complexity of the algorithm for a reasonable range of data is $O(|V|^4)$.

As a final remark in this section, we compare the results summarized in Tables 1 and 2 to the basic optimal LP solution of the MCFP, as stated in the edge path formulation. Algorithm $R$ has the advantage of keeping the number of active paths as low as the number of nonzero variables in the basic optimal LP solution. Therefore it solves larger problems faster. Note that the edge path formulation may require too many variables. This is not the case with $R$. More advanced LP techniques which use special network structures are used to solve the MCFP [He83]. However, even these techniques do not solve problems of size 40 vertices or more [PA84]. These observations plus the fact that the LP formulation of a problem usually
ignores the existing structure of the problem support the fact that our technique to solve the 
MCFP is superior to that of LP.

Section 3.5 Applications of the MCFP to the SCP

We stated in Chapter 2 that the sparsest cut problem is NP-hard. It is a surprising result 
that the MCFP can identify a sparsest cut in many instances of the sparsest cut problem. Table 
1 indicates that for 13 out of 20 cases, the sharpest bound on the throughput $Z$ is provided by 
the cut upper bound. We claim that the cut upper bound in all these cases corresponds to the 
density of a sparsest cut in $G$, where this sparsest cut is identified in step 2 of $R$. To show the 
correctness of our claim we use the finite precision rational arithmetic. The Farey series $F_n$ of 
order $n$ is the ascending series of irreducible fractions between 0 and 1 whose denominators do not exceed $n$ [HW79]. Therefore

$$0 \leq h \leq k \leq n, \quad (h,k) = 1$$

implies $\frac{k}{h} \in F_n$. As an example

$$F_6 = \{0, 1, 1, 1, 2, 1, 3, 2, 3, 4, 1\}.$$

**Theorem 3.2.** If $\frac{k}{h}, \frac{k'}{h'}$ are two successive terms of $F_n$, then

(a) $kh' - hk' = 1$ [HW79].
(b) $|\frac{k}{h} - \frac{k'}{h'}| \leq \frac{1}{n(n-1)}$ [MK85].

**Theorem 3.3.** Let $G, D$, and $C$ be the input to $R$, where $D$ and $C$ are unit functions. Assume 
that $R$ converges within some $\epsilon$, producing a concurrent flow of throughput $Z$ and a cut $(A, \bar{A})$ 
in $G$ such that $\text{den}(A, \bar{A}) - Z \leq \frac{1}{n^2}$, where $n = \lfloor \frac{|V|}{2} \rfloor \lfloor \frac{|V|}{2} \rfloor$. Then $(A, \bar{A})$ is a sparsest cut in $G$.

**Proof.** Let $(B, \bar{B})$ be any cut in $G$. Observe that $|\langle B, \bar{B} \rangle| = |B||\bar{B}| \leq \lfloor \frac{|V|}{2} \rfloor \lfloor \frac{|V|}{2} \rfloor = n$, implying that

$$\text{den}(B, \bar{B}) \in F_n.$$
Now assume that \((A, \bar{A})\) is not the sparsest cut in \(G\). Then there exists a cut \((C, \bar{C})\) such that \(\text{den}(C, \bar{C}) < \text{den}(A, \bar{A})\). By the observation we just made, \(\text{den}(A, \bar{A}) \in \mathcal{F}_n\) and \(\text{den}(C, \bar{C}) \in \mathcal{F}_n\).

By part (b) of Theorem 3.2, \(\text{den}(A, \bar{A}) - \text{den}(C, \bar{C}) \geq \frac{1}{n(n-1)} > \frac{1}{n^2}\) however

\[
\text{den}(A, \bar{A}) - Z \leq \frac{1}{n^2}.
\]

Therefore

\[
\text{den}(A, \bar{A}) - \text{den}(C, \bar{C}) > \text{den}(A, \bar{A}) - Z,
\]

or \(Z > \text{den}(C, \bar{C})\), which contradicts Lemma 2.2. Thus \((A, \bar{A})\) is a sparsest cut. Q.E.D.

Consider Table 1 and observe that the difference between the cut upper bound and the throughput \(Z\) is less than \(\frac{1}{n^2}\), where \(n = \lfloor \frac{|V_1|}{2} \rfloor \cdot \lfloor \frac{|V_2|}{2} \rfloor\) for all cases with no asterisks except for the graph 8. By Theorem 3.3 the cut upper bound in all such cases is the density of a sparsest cut in \(G\). For the graph 8 the density of a sparsest cut was directly computed, verifying that that cut identified by the MCFP is a sparsest cut.

The fact that the solution to the MCFP can identify a solution to some of the instances of the sparsest cut problem will be used in Chapter 7, where we discuss the applications of the MCFP in cluster analysis. In Chapter 7 we will also show that for all cases in Table 2 the MCFP identifies a sparsest cut as well.
CHAPTER IV

AN \( \varepsilon \)-APPROXIMATION ALGORITHM

In this chapter we present a new version of Algorithm \( R \) to solve the MCFP with uniform capacity. We will prove the correctness and also analyze the complexity of this new algorithm.

Algorithm \( H \) is an \( \varepsilon \)-approximation algorithm [PS82]. The input to the algorithm is \( G, D, \) constant capacity function \( C : E \to c^*, \) and \( \varepsilon, 1 > \varepsilon > 0. \) The output is a concurrent flow \( f \) of throughput \( Z \) and a distance function \( d \) such that \( \frac{Z - Z}{Z} \leq \varepsilon \)

Algorithm \( H \)

H1. [Determine the initial feasible solution.] Set \( n=0. \) For each pair of distinct vertices \( i, j \in V, \) assign \( D'(i,j) = \sum_{k \in S} \frac{D(i,k)}{D(k,j)} \) units of flow to one shortest path \( p \in P_i. \) Calculate the value of \( f_n(e), \) which is equal to \( \sum_{p \in P_i} f_n(p) \) for all \( e \in E. \)

H2. [Find shortest paths and upper bounds on the throughput.] Let \( n=n+1. \) Calculate the distance for each edge using

\[
\delta N (e) = e^{\varepsilon} \left( f_n(e) \right) \sum_{e' \in E} e^{\varepsilon} f_n(e')
\]

Find the shortest paths (one path per pair).

H3. [Reroute the flow and update.] Determine a path \( p^* \) and a shortest path \( p_* \) with the same end vertices such that

\[
f_n(p^*)(d_n(p^*) - d_n(p_*)) \]

is maximised over all pairs of vertices. Let \( P' \) denote the set of all active paths for the flow constructed in this iteration. If \( f_n(p^*)(d_n(p^*) - d_n(p_*)) \leq f_n(p^*) \) then go to \( H_4. \) If \( f_n(p^*)(d_n(p^*) - d_n(p_*)) > f_n(p^*) \), then reroute

\[
\sigma = \frac{\varepsilon}{2|E|} \log \frac{d_n(p_2)}{d_n(p_1)}
\]
units of flow from $p^*$ to $p_*$, where $p_2, p_1$ are disjoint portions of $p^*$, respectively with $d_n(p_2) = \sum_{e \in p_2} e_n(e)$ and $d_n(p_1) = \sum_{e \in p_1} e_n(e)$. If $\sigma > f_n(p^*)$, then reroute all the flow on $p^*$ to $p_*$ and remove path $p^*$ from the list of paths carrying flow between pair $ij$. Update the flow on paths $p_*, p^*$ and the corresponding edges in $p_2, p^*$. Go to $H_2$.

H4. [Compute the throughput and Halt.] Calculate the normalized throughput realized by this solution using

$$Z = \frac{c^*}{f_n^* (\sum_{i < j} D(i, j))},$$

where $f_n^* = \text{Max}\{f_n(e) | e \in E\}$. Set

$$f(p) = \frac{c^* f_n(p)}{f_n^*} \quad \text{for all active paths } p$$

and

$$d(e) = \frac{d_n(e) f_n^*}{c^*} \quad \text{for all } e \in E.$$

**Theorem 4.1.** Algorithm $H$ always terminates. Furthermore, upon termination, $H$ constructs a concurrent flow $f$ and a distance function $d$ such that for all distinct $i, j \in V$,

$$f(p)(d(p) - d(i, j)) \leq \frac{\epsilon}{|P^i|},$$

where $P^i$ is the set of active paths under $f$, and $p$ is any $ij$ path.

**Proof.** Let $S_n$ denote $\sum_{e \in E} e \frac{|E|}{p^*} f_n(e)$ where $f_n$ is the concurrent flow constructed at iteration $n$ in $H_3$. The idea behind the proof is to show that, if $H$ does not halt at iteration $n$, then $S_n - S_{n+1}$ is always greater than a positive constant. Consider $p^*, p_*, p_1, p_2$ as they are defined in $H_3$. Observe that the flow rerouting can only change flow on edges in $p_2$ or $p_1$. Therefore

$$S_n - S_{n+1} = \sum_{e \in p_2} e \frac{|E|}{p^*} f_n(e) + \sum_{e \in p_1} e \frac{|E|}{p^*} f_n(e) - \sum_{e \in p_2} e \frac{|E|}{p^*} f_{n+1}(e) - \sum_{e \in p_1} e \frac{|E|}{p^*} f_{n+1}(e). \quad (4.1)$$

Recall from $H_3$ that the amount of flow to reroute is $\sigma = \frac{\epsilon}{|P^i|} \log \frac{d_n(p_2)}{d_n(p_1)}$. Now consider the following two cases.
Case 1. $\sigma \leq f_n(p^*)$. Then using (4.1), we get
\[
\frac{S_n - S_{n+1}}{S_n} = \frac{\sum_{e \in \mathcal{E}(p^*)} e^{\frac{1}{\tau} f_n(e)}}{S_n} + \frac{\sum_{e \in \mathcal{E}(p)} e^{\frac{1}{\tau} f_n(e)}}{S_n} - \frac{\sum_{e \in \mathcal{E}(p^*)} e^{\frac{1}{\tau} f_{n+1}(e)}}{S_n} - \frac{\sum_{e \in \mathcal{E}(p)} e^{\frac{1}{\tau} f_{n+1}(e)}}{S_n}.
\]

Considering the fact that $f_{n+1}(e) = f_n(e) + \sigma$ for all $e \in p_1$, and $f_{n+1}(e) = f_n(e) - \sigma$ for all $e \in p_2$, we get
\[
\frac{S_n - S_{n+1}}{S_n} = \frac{\sum_{e \in \mathcal{E}(p_2)} e^{\frac{1}{\tau} (f_n(e) - \sigma)}}{S_n} + \frac{\sum_{e \in \mathcal{E}(p_1)} e^{\frac{1}{\tau} (f_n(e) + \sigma)}}{S_n} - \frac{\sum_{e \in \mathcal{E}(p_2)} e^{\frac{1}{\tau} (f_{n+1}(e) - \sigma)}}{S_n} - \frac{\sum_{e \in \mathcal{E}(p_1)} e^{\frac{1}{\tau} (f_{n+1}(e) + \sigma)}}{S_n}.
\]

However, $e^{\frac{1}{\tau} \sigma} = \sqrt{\frac{d_n(p_2)}{d_n(p_1)}}$. Therefore (4.2) implies
\[
\frac{S_n - S_{n+1}}{S_n} = d_n(p_2) + d_n(p_1) - e^{-\frac{1}{\tau} \sigma} d_n(p_2) - e^{-\frac{1}{\tau} \sigma} d_n(p_1).
\]

Thus
\[
\frac{S_n - S_{n+1}}{S_n} = \left(\frac{d_n(p_2) - d_n(p_1)}{\sqrt{d_n(p_2)} + \sqrt{d_n(p_1)}}\right)^2.
\]

But $p^*$ and $p_*$ are involved in flow rerouting; thus $f_n(p^*)(d_n(p^*) - d_n(p_*)) > \frac{1}{|p^*|}$. Therefore,
\[
d_n(p_2) - d_n(p_1) = d_n(p^*) - d_n(p_*) > \frac{1}{f_n(p^*)|p^*|}.\]

Notice that $f_n(p^*) \leq \sum_{i<j} D'(i,j) = 1$. Hence
\[
d_n(p_2) - d_n(p_1) > \frac{1}{|p^*|}.\]

Observe that $d_n(p_2)$ and $d_n(p_1)$ are bounded above by $\sum_{e \in E} d_n(e) = 1$. This is used together with (5.1) to get a lower bound for $\frac{S_n - S_{n+1}}{S_n}$ in (4.3). Therefore
\[
\frac{S_n - S_{n+1}}{S_n} \geq \left(\frac{\epsilon}{|p^*|}\right)^2.
\]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Case 2. $\sigma > f_n(p^*)$. Then all the flow on $p^*$ is rerouted to $p_*$. Applying the same technique as for case 1, we get

$$\frac{S_n - S_{n+1}}{S_n} = d_n(p_2) + d_n(p_1) - e^{-\frac{|E|}{|P|}}f_n(p^*)d_n(p_2) - e^{-\frac{|E|}{|P|}}f_n(p^*)d_n(p_1).$$

Now consider the function

$$F(x) = d_n(p_2) + d_n(p_1) - e^{-\frac{|E|}{|P|}}d_n(p_2) - e^{-\frac{|E|}{|P|}}d_n(p_1).$$

Then $(F(x))' = \frac{|E|}{|P|}(e^{-\frac{|E|}{|P|}}d_n(p_2) - e^{-\frac{|E|}{|P|}}d_n(p_1))$. Thus,

$$(F(x))' \geq 0, \quad \text{for} \quad 0 \leq x \leq \frac{\epsilon}{2|E|} \log \frac{d_2(p)}{d_1(p)} = \sigma. \quad (4.7)$$

However, $f_n(p^*)(d_n(p^*) - d_n(p_*)) > \frac{|E|}{|P|}$ implies that $f_n(p^*) > |\frac{E}{P}|$. Also, (4.7) implies that $F$ is monotonically increasing in $[0, \sigma]$. Thus

$$F(f_n(p^*)) = \frac{S_n - S_{n+1}}{S_n} > \frac{\epsilon}{|P|} d_n(p_2) + d_n(p_1) - e^{-\frac{|E|}{|P|}}d_n(p_2) - e^{-\frac{|E|}{|P|}}d_n(p_1)$$

$$= \left(\frac{e^{-\frac{|E|}{|P|}} - 1}{e^{-\frac{|E|}{|P|}}} \right)(d_n(p_2) - e^{-\frac{|E|}{|P|}}d_n(p_1)). \quad (4.8)$$

Note that $f_n(p^*) < \sigma$ and $f_n(p^*) > |\frac{E}{P}|$ together imply $|\frac{E}{P}| < \sigma$. Substituting the value of $\sigma$, we get $|\frac{E}{P}| < \frac{\epsilon}{2|E|} \log \frac{d_2(p_2)}{d_1(p_1)}$, which implies $d_n(p_1) < \frac{d_2(p_2)}{e^{-|E|/|P|}}$. Substituting this upper bound for $d_n(p_1)$ in (4.8) yields

$$\frac{S_n - S_{n+1}}{S_n} \geq d_n(p_2)(\frac{e^{-\frac{|E|}{|P|}} - 1}{e^{-\frac{|E|}{|P|}}})^2. \quad (4.9)$$

Also, $f_n(p^*)(d_n(p_2) - d_n(p_1)) > |\frac{E}{P}|$ and $f_n(p^*) \leq 1$ together imply that $d_n(p_2) > |\frac{E}{P}|$. Thus (4.9) results in

$$\frac{S_n - S_{n+1}}{S_n} > \frac{\epsilon}{|P|}(\frac{e^{-\frac{|E|}{|P|}} - 1}{e^{-\frac{|E|}{|P|}}})^2 = \frac{\epsilon}{|P|}(1 - e^{-\frac{|E|}{|P|}})^2.$$

But $|P| \geq \left(\frac{|P|}{2}\right) \geq |E|$, implying that $\frac{|E|}{|P|} \leq 1$. It follows that (using exponential expansion)

$$1 - e^{-\frac{|E|}{|P|}} \geq \frac{|E|}{|P|} - \frac{|E|^2}{2|P|^2} = \frac{|E|}{|P|}(1 - \frac{|E|}{2|P|})^2.$$
or

\[
\frac{S_n - S_{n+1}}{S_n} \geq \frac{\varepsilon |E|^2}{|P|^3} (1 - \frac{|E|}{2|P|^3})^2.
\]

Considering that \(\frac{|E|}{|P|^3} \leq 1\), we get

\[
\frac{S_n - S_{n+1}}{S_n} \geq \frac{\varepsilon |E|^2}{4|P|^3}.
\]

Observe that \(|P| \leq |V|, |P'| \leq |P|\), and \(\varepsilon \leq 1\). Therefore in either Case 1 or Case 2,

\[
\frac{S_n - S_{n+1}}{S_n} \geq \frac{\varepsilon^2}{4|P|^3} \geq \frac{\varepsilon^2}{4(|V|)^3}.
\]

Now assume that \(\mathcal{R}\) did not halt after \(m\) iterations. Then

\[
\frac{S_i - S_{i+1}}{S_i} \geq \frac{\varepsilon^2}{4(|V|)^3}, \quad \text{for all } i, 1 \leq i \leq m.
\]

Thus

\[
\frac{S_{i+1}}{S_i} \leq 1 - \frac{\varepsilon^2}{4(|V|)^3}, \quad \text{for all } i, 1 \leq i \leq m.
\]

This implies

\[
\frac{S_{m+1}}{S_1} \leq (1 - \frac{\varepsilon^2}{4(|V|)^3})^m.
\]

However, \(S_1 \leq |E| e^{\frac{|E|}{\varepsilon}}\) and \(S_{m+1} \geq |E|\). Therefore

\[
e^{\frac{|E|}{\varepsilon}} \leq (1 - \frac{\varepsilon^2}{4|V|^3})^m
\]

or

\[
m \leq \frac{|E|}{\varepsilon \ln(1 + \frac{\varepsilon^2}{4|V|^3})}.
\]

Therefore \(\mathcal{H}\) halts. Assume \(n = N\) on termination. Then \(f(p) = \frac{\epsilon^x}{f_N} f_N(p)\) for all \(p \in P\).

Also, \(\mathcal{D}(\varepsilon) = \frac{\epsilon^x}{\varepsilon} d_N(\varepsilon)\) for all \(\varepsilon \in E\). Therefore

\[
f(p)(\mathcal{D}(p) - \mathcal{D}(i, j)) = f_N(p)(d_N(p) - d_N(i, j)) \leq \frac{\varepsilon}{|P'|} \text{ for any } ij \text{ path } p. \tag*{Q.E.D.}
\]
Lemma 4.2. Let $\epsilon > 0$. Assume $f_n$, $d_n$ are the concurrent flow and the distance function constructed by $H$ at iteration $n$, $n > 0$. Then

$$\sum_{p \in P} f_n(p)d_n(p) + \epsilon \geq f^*_n$$

where $f^*_n = \max\{f_n(e) | e \in E\}$.

Proof. Observe that

$$\sum_{p \in P} f_n(p)d_n(p) = \sum_{e \in E} f_n(e)d_n(e);$$

therefore

$$f^*_n - \sum_{p \in P} f_n(p)d_n(p) = f^*_n - \sum_{e \in E} f_n(e)d_n(e).$$

But $\sum_{e \in E} d_n(e) = 1$. Thus

$$f^*_n - \sum_{p \in P} f_n(p)d_n(p) = f^*_n \sum_{e \in E} d_n(e) - \sum_{e \in E} f_n(e)d_n(e) =$$

$$\sum_{e \in E} (f^*_n - f_n(e))d_n(e) = \sum_{e \in E} (f^*_n - f_n(e)) \frac{e^{\|t\|f_n(e)}}{\sum_{e' \in E} e^{\|t\|f_n(e')}}.$$

(4.11)

However,

$$\frac{e^{\|t\|f_n(e)}}{\sum_{e' \in E} e^{\|t\|f_n(e')}} \leq \frac{e^{\|t\|f_n(e)}}{e^{\|t\|f^*_n}} = \frac{1}{e^{\|t\| (f^*_n - f_n(e))}}.$$

Therefore (4.11) implies that

$$f^*_n - \sum_{p \in P} f_n(p)d_n(p) \leq \sum_{e \in E} \frac{f^*_n - f_n(e)}{e^{\|t\| (f^*_n - f_n(e))}}.$$

But $f^*_n - f_n(e) \geq 0$. It follows that $e^{\|t\| (f^*_n - f_n(e))} \geq |E| (f^*_n - f_n(e))$ (using exponential expansion).

Therefore

$$\sum_{e \in E} \frac{f^*_n - f_n(e)}{e^{\|t\| (f^*_n - f_n(e))}} \leq \sum_{e \in E} \frac{f^*_n - f_n(e)}{e^{\|t\| (f^*_n - f_n(e))}} = \sum_{e \in E} \epsilon = \epsilon,$$

implying $f^*_n - \sum_{p \in P} f_n(p)d_n(p) \leq \epsilon$, which proves the lemma. Q.E.D

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Theorem 4.3. Let $\epsilon > 0$, and define $\epsilon' = (\frac{4}{|E|}[c_{(c+1)}])$. Assume $G$, $D$, the constant capacity function $C : E \rightarrow c^*$, and $\epsilon'$ are the input to $H$. Then after at most

$$\frac{2|E|^2(|E| + \epsilon)}{\epsilon \ln(1 + \frac{|E|^2}{18|E|^2([|E| + \epsilon]^2([|V|]^2)})}$$

iterations,

(a) $Z \geq \sum_{i<j} d(i,j)C(i,j) - \sum_{i<j} c_{*,} D(i,j)$ and

(b) $\frac{Z - \epsilon}{2} \leq \epsilon$,

where $d$ is the distance function and $Z$ is the throughput of the concurrent flow $f$ constructed by $H$ upon termination.

Proof of (a) By Theorem 4.1, Algorithm H terminates after at most

$$\frac{2|E|^2(|E| + \epsilon)}{\epsilon \ln(1 + \frac{|E|^2}{18|E|^2([|E| + \epsilon]^2([|V|]^2)})}$$

iterations. Denote the number of iterations to terminate by $m$. Let $P'$ denote the set of active paths upon termination. Then

$$\sum_{p \in P} f(p)d(p) = \sum_{P' \in P} f(p)d(p) = \sum_{i<j} \sum_{P \in P'} f(p)d(p).$$

But $f$ and $d$ are constructed upon termination. Therefore $f(p)(d(p) - d(i,j)) \leq \frac{\epsilon'}{|P'|}$ for any path $p \in P_{ij}$. Thus (4.12) implies

$$\sum_{p \in P'} f(p)d(p) \leq \sum_{i<j} \sum_{P \in P_{ij} \cap P'} f(p)(d(i,j) + \epsilon') = \sum_{i<j} \sum_{P \in P_{ij} \cap P'} f(p)d(i,j) + |P'|\frac{\epsilon'}{|P'|}$$

$$= \sum_{i<j} d(i,j) \sum_{P \in P_{ij} \cap P'} f(p) + \epsilon' = \sum_{i<j} \sum_{P \in P_{ij} \cap P'} f(p)d(i,j)D'(i,j) + \epsilon'.$$

Also, by Lemma 4.2, $\sum_{p \in P} f_m(p)d_m(p) \geq f_*^m - \epsilon'$. Notice that

$$\sum_{p \in P} f(p)d(p) = \sum_{p \in P} f_m(p)d_m(p).$$

Hence $\sum_{p \in P} f(p)d(p) \geq f_*^m - \epsilon'$. Combining this and (4.13) we have

$$\frac{\epsilon^*}{f_m} \sum_{i<j} d(i,j)D'(i,j) + \epsilon' \geq f_*^m - \epsilon'.$
or

\[
\frac{c^*}{f_m} \sum_{i<j} d(i,j) D'(i,j) + 2\varepsilon' \geq f_m. \tag{4.14}
\]

Substituting \( f_m \) by \( \frac{c^*}{\sum_{i<j} D(i,j)} \) on the right hand side of (4.14) implies

\[
(\sum_{i<j} D(i,j))\left( \frac{c^*}{f_m} \sum_{i<j} d(i,j) D'(i,j) + 2\varepsilon' \right) \geq c^*. 
\]

But \( \sum_{e \in \mathcal{E}} f_m(e) \geq \sum_{i<j} D'(i,j) = 1 \). Therefore \( f_m \geq \frac{1}{|\mathcal{E}|} \), implying \( Z \leq c^* \frac{|\mathcal{E}|}{\sum_{i<j} D(i,j)} \). Using this upper bound for \( Z \) in the previous inequality, we have

\[
\frac{c^*}{f_m} \sum_{i<j} d(i,j) D(i,j) + c^* 2\varepsilon' |\mathcal{E}| \geq c^*
\]
or

\[
Z \geq \frac{f_m}{\sum_{i<j} d(i,j) D(i,j)} - \frac{2 f_m \varepsilon' |\mathcal{E}|}{(\sum_{i<j} D(i,j)) \sum_{i,j} d(i,j) D'(i,j)}. \tag{4.15}
\]

Now use \( f_m \geq \frac{1}{|\mathcal{E}|} \) in (4.14). Then

\[
2\varepsilon' + \frac{c^*}{f_m} \sum_{i<j} d_n(i,j) D'(i,j) \geq \frac{1}{|\mathcal{E}|}
\]
or

\[
\sum_{i<j} d(i,j) D'(i,j) \geq \frac{f_m}{c^*} \left( \frac{1}{|\mathcal{E}|} - 2\varepsilon' \right).
\]

Replace the value of \( \varepsilon' \) by \( \frac{c^*}{|\mathcal{E}| + c^*} \) in the last inequality. Then

\[
\sum_{i<j} d(i,j) D'(i,j) \geq \frac{f_m}{c^*} \left( \frac{1}{|\mathcal{E}|} - \left( \frac{c^*}{|\mathcal{E}| + c^*} \right) \right) = \frac{f_m}{c^* (|\mathcal{E}| + c^*)} > 0.
\]

Substituting \( \frac{f_m}{c^* (|\mathcal{E}| + c^*)} \) for \( \sum_{i<j} d(i,j) D'(i,j) \) and \( \varepsilon' \) by \( \frac{c^*}{|\mathcal{E}| + c^*} \) in (4.15) will give

\[
Z \geq \frac{f_m}{d(i,j) D(i,j)} - \frac{2 f_m \varepsilon' |\mathcal{E}|}{c^* (|\mathcal{E}| + c^*)} = \frac{f_m}{\sum_{i<j} d(i,j) D(i,j)} - \frac{\varepsilon c^*}{\sum_{i<j} D(i,j)}.
\]

But

\[
\sum_{e \in \mathcal{E}} d(e) C(e) = \sum_{e \in \mathcal{E}} \frac{f_m}{c^*} d_m(e) c^* = f_m \sum_{e \in \mathcal{E}} d_m(e) = f_m.
\]
Hence
\[
2 \geq \frac{\sum_{e \in B} d(e)C(e)}{\sum_{i < j} d(i,j)D(i,j)} - \frac{\epsilon \epsilon^*}{\sum_{i < j} D(i,j)}.
\]

Proof of (b) Using part a and lemma 2.3, after \(m\) iterations, we obtain

\[
\frac{\sum_{e \in B} d(e)C(e)}{\sum_{i < j} d(i,j)D(i,j)} \geq \frac{2}{2} \geq \frac{\sum_{e \in B} d(e)C(e)}{\sum_{i < j} d(i,j)D(i,j)} - \frac{\epsilon \epsilon^*}{\sum_{i < j} D(i,j)}.
\]

Therefore, \(2 - \frac{2}{2} \leq \frac{\epsilon \epsilon^*}{\sum_{i < j} D(i,j)}\). But \(2 \geq \frac{\epsilon \epsilon^*}{\sum_{i < j} D(i,j)}\), since an assignment of \(D'(i,j)\) units of flow to the shortest \(ij\) path yields a concurrent flow of throughput \(\frac{\epsilon \epsilon^*}{\sum_{i < j} D(i,j)}\). Thus

\[
\frac{2 - \frac{2}{2}}{2} = \frac{\sum_{i < j} D(i,j)}{\sum_{i < j} D(i,j)} = \epsilon.
\]

Q.E.D

In practice, it is not feasible (or at least difficult) to compute the values of the distance function defined in Algorithm H since the value of \(e_{i,j}^{(m)}\) is usually very large. Therefore, should Algorithm H be implemented, we suggest defining the distance function as \(d(e) = \frac{\alpha e_j^{(m)}}{\sum_{i < j} e_j^{(m)}},\) where the value of \(\alpha\) is small at the beginning and is increased if the convergence rate is slow.

This explains the reason for changing \(\alpha\) in Algorithm R, which is a practical version of H.
CHAPTER V

THE MCFP WITH UNIFORM DEMAND

Let $G$, $D$, and $C$ be an instance of the MCFP such that $D$ is a constant nonzero function. We refer to such an instance of the MCFP as the MCFP with uniform demand. In this chapter we seek the relationship between the MCFP with uniform demand and the MCFP with uniform capacity.

Let $C : E \to \mathbb{R}^+$ be a capacity function on graph $G = \langle V, E \rangle$. We extend $C$ to have the domain $V \times V$ as follows:

$$
C(i,j) = \begin{cases} 
C(e) & \text{for all } i,j \in V \text{ such that } e = ij, \\
0 & \text{otherwise.}
\end{cases}
$$

Notice that the extended capacity function may not be a constant function, even if the original capacity function is a constant function. Through the rest of this section we will use the extended capacity function and assume the underlying graph for the MCFP is a complete graph.

Algorithm Modify on the next page is important. It will be used to construct a reduction from the MCFP with uniform demand to the MCFP with uniform capacity. The input to the algorithm consists of $G$, $C$, $D$, and a maximum concurrent flow function $f$ which is explicitly defined on the set of active paths $P'$. The algorithm modifies $f$ and $P'$ so that the maximum capacity of any edge is utilized to supply the flow between the end vertices of that edge, while maintaining the maximality of the throughput. We will assume through Algorithm Modify that $P_e$ denotes the set of active paths containing the edge $e$, and $P_{ij}$ denotes the set of all active $ij$ paths. This assumption is only made through Algorithm Modify and $P_e$, $P_{ij}$ have their original meaning elsewhere. Also, we define a unit $ij$ path to be a path containing only the edge $ij$. 

30

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Algorithm Modify

/ Given $G, C, D$, and a maximum concurrent flow with the set of active paths $P'$, this algorithm modifies $f$ such that for every unit $ij$ path $p$, $f(p) \geq \min\{2D(i,j), C(i,j)\}$. /

FOR each commodity $ij$

IF $C(i,j) > 0$ THEN

Let $p$ denote the unit $ij$ path with $e = ij$.

WHILE $f(e) < C(i,j)$ AND $f(p) < 2D(i,j)$ / Fill out $e$ to the capacity./

Select an active $ij$ path $p_1$, $p_1 \neq p$ and set $\sigma = \min\{C(i,j) - f(e), f(p_1)\}$.

Reroute $\sigma$ units of flow from $p_1$ to $p$ and update $f(e)$, $f(p_1)$, $f(p)$, and $P_{ij}$, $P_e$ (if necessary).

ENDWHILE

WHILE $P_e \neq \{p\}$ AND $P_{ij} \neq \{p\}$

Select $p_2 \in P_e - \{p\}$. Assume the end vertices for $p_2$ are $t, k$.

Select $q \in P_{ij} - \{p\}$. Construct a $tk$ path $q_1$ such that $q_1$ does not contain $e$ and every edge in $q_1$ is either an edge in $p_2$ or in $q$. Set $\sigma = \min\{f(p_2), f(q)\}$.

Reroute $\sigma$ units of flow from $q$ to $p$. Also reroute $\sigma$ units of flow from $p_2$ to $q_1$ and update $f(q), f(p), f(p_2), f(q_1), P_{tk}$ and $P_e, P_{ij}$ (if necessary).

ENDWHILE

ENDIF

ENDFOR

END

The data structure for this algorithm is the same as the data structure for algorithm $R$.

Note that the doubly-linked list to represent paths containing an edge is effectively used in this algorithm.

Complexity. Suppose that $i$ and $j$ are fixed. The first WHILE statement runs at most $|P_{ij}|$ times. Updation $P_e$ is done in the time complexity $O(|V|)$. Therefore this statement is of
time complexity $O(|V||P_i|)$ (for one commodity). Every time the second WHILE statement is executed, $|P_i|$ or $|P_e|$ is reduced by 1. Therefore this WHILE statement is not executed more than $|P_i| + |P_e|$ times. Notice that Modify never increases the number of active paths since if a new active path is created then another active path is deleted. Therefore $|P_i| + |P_e| \leq |P'|$. It follows that the time complexity for this part is $O(|V|^2|P'|)$. Notice that the construction of $q_1$ is done in $O(|V|^2)$. We thus conclude that the order of time complexity for the algorithm is $O(|V|^4|P'|)$.

The correctness of the algorithm follows from the fact that for any commodity $ij$ the algorithm modifies the flow function appropriately, and while doing this it does not change the flow on any unit $tk$ path, if $\{t,k\} \neq \{i,j\}$ (i.e., it does not undo what it has done before).

An important observation (which was also used in the analysis of the complexity) is that the number of active paths in the output is at most $|P'|$. We now summarize our results regarding Algorithm Modify in a lemma.

**Lemma 5.1.** Let $G, D, C$ be an instance of the MCFP. Then there exists a maximum concurrent flow function $\hat{f}$ of throughput $2$ such that for every unit $ij$ path $p$

$$\hat{f}(p) \geq \min\{C(i,j), 2D(i,j)\},$$

for all $i,j \in V$. Furthermore, $\hat{f}$ can be obtained from any maximum concurrent flow function $f$ with the set of active paths $P'$ in time complexity $O(|V|^4P')$ such that the number of active paths under $\hat{f}$ is at most $|P'|$.

A nontrivial instance of the MCFP with uniform demand consists of $G, D, C$ such that the extended capacity function $C$ is not a constant function. We now present the main result for this section.

**Theorem 5.2.** Let $G = < V, E >$, $D, C$ be a nontrivial instance of the MCFP with uniform demand and a maximum concurrent flow $\hat{f}$ (of throughput $2$) with

$$D : V \times V \rightarrow D_0,$$
for some \( D_0 \in \mathbb{R}^+ \). Denote by \( G, D', C' \), an instance of the MCFP with uniform capacity having a maximum concurrent flow \( f \) (of the throughput \( Z \)) such that

\[
D'(i,i) = 0, D'(i,j) = c^* - C(i,j) \quad \text{for all } i, j \in V,
\]

where \( c^* = \max_{i,j \in V} \{C(i,j)\} \) and \( C' : V \times V \to D_0 \). Then

\[
Z = \frac{c^*}{D_0} - \frac{1}{Z}.
\]

Furthermore, \( f_2 \) defined on \( P \) as

\[
f_2(p) = \begin{cases} 
\frac{f(p)}{D_0}, & \text{if } p \text{ is unit } ij \text{ path for some } i, j \in V, \\
\frac{f(p)}{Z}, & \text{otherwise,}
\end{cases}
\]

is a maximum concurrent flow for \( G, D, \) and \( C \).

**Proof.** Using the extended capacity functions \( C \) and \( C' \) we may assume that \( G \) is complete. Therefore, \( \hat{f} \) is the solution to

\[
\begin{align*}
\text{Maximize } Z & \text{ subject to } \\
\sum_{p \in P_{ij}} f(p) &= D_0 Z \\
\sum_{p \in P_{i,j}} f(p) &\leq C(i,j) \quad \text{for all } e \in E, i < j, \text{ with } e = ij, \\
f(p) &\geq 0 \quad \text{for all } p \in P.
\end{align*}
\]

(5.1)

Similarly \( \hat{f} \) is the optimal solution to

\[
\begin{align*}
\text{Maximize } Z & \text{ subject to } \\
\sum_{p \in P_{j,i}} f(p) &= Z(c^* - C(i,j)) \quad \text{for all } i, j \in V, i < j, \\
\sum_{p \in P_{i,j}} f(p) &\leq D_0 \quad \text{for all } e \in E, \\
f(p) &\geq 0 \quad \text{for all } p \in P.
\end{align*}
\]

(5.2)

Assume, with no loss of generality, that \( \hat{f} \) and \( \hat{f} \) are obtained using Lemma 5.1. Since \( C \) is not a constant function, Lemma 2.2 implies that \( \frac{c^*}{D_0} > \hat{Z} \) or \( c^* > D_0 \hat{Z} \). Define \( f_1 \) on \( P \) as

\[
f_1(p) = \begin{cases} 
D_0 \left( \hat{f}(p) + c^* - C(i,j) - D_0 \hat{Z} \right) / c^* - D_0 \hat{Z}, & \text{if } p \text{ is a unit } ij \text{ path for some } i, j \in V, \\
D_0 \left( \hat{f}(p) / c^* - D_0 \hat{Z} \right), & \text{otherwise.}
\end{cases}
\]

Then

\[
\sum_{p \in P_{ij}} f_1(p) = D_0 \sum_{p \in P_{ij}} \hat{f}(p) / c^* - D_0 \hat{Z} + D_0 \left( c^* - D_0 \hat{Z} - C(i,j) / c^* - D_0 \hat{Z} \right) =
\]
\[
\frac{D_0^2 \hat{Z}}{c^* - D_0 \hat{Z}} + \frac{D_0 c^* - D_0^2 \hat{Z} - D_0 \hat{Z}}{c^* - D_0 \hat{Z}} c^* - D_0 \hat{Z} (c^* - C(i,j))^t \quad \text{for all } i, j \in V, i < j.
\]

(5.3)

Also,
\[
\sum_{p \in P_e} f_1(p) = \frac{D_0 \sum_{p \in P_e} \hat{f}(p) + D_0 (c^* - \hat{Z} - C(i,j))}{c^* - D_0 \hat{Z}} \leq \frac{D_0 C(i,j) + D_0 c^* - D_0 \hat{Z} - D_0 C(i,j)}{c^* - D_0 \hat{Z}} = D_0 \quad \text{for all } e \in E.
\]

We now show that \( f_1(p) \geq 0 \) for all \( p \in P \), implying that \( f_1 \) is a concurrent flow for (5.2). It suffices to show for every unit \( ij \) path \( p \), \( f_1(p) \geq 0 \). Let \( p \) be any unit \( ij \) path for some \( i, j \in V \), then
\[
f_1(p) = \frac{D_0 (\hat{f}(p) + c^* - C(i,j) - D_0 \hat{Z})}{c^* - D_0 \hat{Z}}.
\]

By Lemma 5.2, \( \hat{f}(p) \geq \text{Min}\{C(i,j) + D_0 \hat{Z}\} \). Therefore
\[
f_1(p) \geq \frac{D_0 \{\text{Min}\{C(i,j) + D_0 \hat{Z}\} + c^* - C(i,j) - D_0 \hat{Z}\}}{c^* - D_0 \hat{Z}} = \frac{D_0 \{\text{Min}\{C(i,j) + c^* - C(i,j) - D_0 \hat{Z}, D_0 \hat{Z} + c^* - C(i,j) - D_0 \hat{Z}\}\}}{c^* - D_0 \hat{Z}}.
\]

By Lemma 2.2, \( \frac{c^*}{D_0} > \hat{Z} \) or \( c^* > D_0 \hat{Z} \). Also, \( c^* \geq C(i,j) \) for all \( i, j \in V \). Thus \( f_1(p) \geq 0 \), implying that \( f_1 \) is a concurrent flow for (5.2). Notice that (5.3) now implies
\[
\hat{Z} \geq \frac{D_0}{c^* - D_0 \hat{Z}}.
\]

(5.4)

Conversely, consider the instance of the MCFP with uniform capacity. It is possible to assign \( \frac{D_0 D'(i,j)}{\sum_{i < j} D'(i,j)} \) units of flow on each edge between every pair, constructing a concurrent flow of throughput \( \frac{D_0}{\sum_{i < j} D'(i,j)} \). Notice that \( \sum_{i < j} D'(i,j) = \sum_{i < j} (c^* - C(i,j)) > 0 \), since \( C \) is not a constant function. Therefore \( \hat{Z} \geq \frac{D_0}{\sum_{i < j} D'(i,j)} > 0 \) which implies that \( f_2 \) is well defined. Furthermore,
\[
\sum_{p \in P_{ij}} f_2(p) = \frac{\sum_{p \in P_{ij}} \hat{f}(p)}{\hat{Z}} + C(i,j) - \frac{D_0}{\hat{Z}} = \frac{\hat{Z}(c^* - C(i,j))}{\hat{Z}} + C(i,j) - \frac{D_0}{\hat{Z}} =
\]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ \frac{2c^* - 2C(i,j) + 2C(i,j) - D_0}{2} = \frac{2c^* - D_0}{2} = \]
\[ \left( \frac{c^*}{D_0} - \frac{1}{2} \right)D_0 \quad \text{for all } i, j \in V, i < j. \]

Also,
\[ \sum_{p \in P_2} f_2(p) = \sum_{p \in P_2} \frac{f(p)}{2} + C(i,j) - \frac{D_0}{2} \leq \]
\[ \frac{D_0}{2} + C(i,j) - \frac{D_0}{2} = C(i,j) \quad \text{for all } e \in E \text{ with } e = ij. \]

Now observe that for any nonunit \( ij \) path \( p \), \( f_2(p) = \frac{f(p)}{2} \geq 0 \). Let \( p \) be any unit \( ij \) path, then
\[ f_2(p) = \frac{f(p) + 2C(i,j) - D_0}{2}. \]

By Lemma 5.2
\[ f(p) \geq \min \{ 2(c^* - C(i,j)), D_0 \}, \]

thus
\[ f_2(p) \geq \frac{\min \{ 2(c^* - C(i,j)), D_0 \} + 2C(i,j) - D_0}{2} = \frac{\min \{ 2c^* - D_0, 2C(i,j) \}}{2}. \]

But \( 2C(i,j) \geq 0 \). Also, \( 2c^* \geq D_0 \) (since an assignment of \( D_0 \frac{c^* - C(i,j)}{c^*} \) units of flow to each unit \( ij \) path \( p \), for all \( i, j \in V, i < j \), will produce a concurrent flow for (5.1) with the throughput of \( \frac{D_0}{c^*} \), thus \( 2 \geq \frac{D_0}{c^*} \)). It follows that \( f_2 \) is a concurrent flow for (5.1) of throughput
\[ \frac{c^*}{D_0} - \frac{1}{2}. \]

Therefore
\[ \hat{Z} \geq \frac{c^*}{D_0} - \frac{1}{2} \]

or
\[ \hat{Z} \leq \frac{D_0}{c^* - D_0 \hat{Z}}. \tag{5.5} \]

Notice that (5.4) and (5.5) together imply
\[ \hat{Z} = \frac{c^*}{D_0} - \frac{1}{2}. \]
It follows that $f_2$ is a maximum concurrent flow for (5.1) which proves the theorem. Q.E.D.

Note that Lemma 5.1 and Theorem 5.2 together identify a reduction from the MCFP with uniform demand to the MCFP with uniform capacity.
CHAPTER VI

THE MCFP AND THE SINGLE COMMODITY FLOW PROBLEM

Section 6.1 The Single Commodity Flow Problem

In this chapter we investigate the similarities between the MCFP and the single commodity flow problem. The single commodity flow problem [FF62] has a rich and elegant theory and many applications in operations research and combinatorics. The original flow augmentation algorithm developed by Ford and Fulkerson [FF62] has all of the following features: (a) identification of a maximum flow value and flow assignment achieving that flow, (b) identification of a minimum capacity cut which contains the edges saturated under any optimal solution, and (c) a proof of the max-flow min-cut theorem. The elegance of this approach was in the understanding of the structure of the underlying problem provided by the interpretation of the dual problem (the min-cut problem) that is implicitly solved. The original labeling algorithm to solve the problem in [FF62] was not polynomially time bounded, but a series of faster and faster algorithms have been devised for the problem (see [TA83]). Note that the single commodity problem is a special case of the MCFP where the demands are assumed to be zero for all pairs except for one pair, for which the demand is one. Let $u, v \in V$. Then $(B, \bar{B})$ is a $uv$ separating cut if $u$ and $v$ are not both in $B$ nor both in $\bar{B}$. The following is a new proof for the undirected version of the max-flow min-cut theorem, using the rich structure of the MCFP.

Theorem 6.1. Let $G, D, C$ be an instance of the MCFP with two distinct vertices $u, v \in V$ and a maximum concurrent flow of throughput $\hat{Z}$, such that $D$ is defined as

$$D(i, j) = \begin{cases} 0 & \text{if } \{i, j\} \neq \{u, v\} \\ 1 & \text{otherwise.} \end{cases}$$
Then
\[ \mathcal{Z} = C(A, \bar{A}), \]
where \((A, \bar{A})\) is a minimum capacity uv separating cut.

**Proof.** Let \(\tilde{f}\) be a maximum concurrent flow saturating only the critical edges in \(G\) as in Theorem 2.6. Denote the MCF-partition by \((A_1, A_2, \ldots, A_k)\), for some \(k \geq 2\). Assume with no loss of generality that \(u \in A_1\). Then \(v \in A_j, j \neq 1\), for otherwise there exists an active uv path \(p\) in \(A_1\) such that none of the edges in \(p\) is saturated. Also, since \(u, v\) are the only vertices with no zero demand, there exists a uv path \(p'\) with at least one critical edge in it. Then some amount of flow can be rerouted from \(p'\) to \(p\), maintaining the optimality of the throughput and contradicting the criticality of \(e\); therefore \(j \neq 1\). Now observe that no active uv path \(p\) can cross the cut \((A_1, \bar{A}_1)\) more than once. For if this is true, then there is a \(u_1u_2\) subpath \(p'\) of \(p\) for some \(u_1, u_2 \in A_1\), which does not contain any of the edges in \(E(\langle A_1 \rangle)\). Since the endvertices of \(p'\) are in \(A_1\), it must contain some critical edges. Let \(q\) be a \(u_1u_2\) path in \(A_1\). Construct a uv path using \(p\) by replacing the subpath \(p'\) with \(q\). Then some amount of flow can be rerouted from \(p\) to this new path, maintaining the optimality of the throughput and contradicting the fact that some edge in \(p'\) must be critical. Thus every active uv path contains exactly one edge from the cut \((A_1, \bar{A}_1)\), implying
\[ \mathcal{D}(A_1, \bar{A}_1) = C(A_1, \bar{A}_1). \]

But \(D(A_1, \bar{A}_1) = 1\); thus
\[ \mathcal{Z} = \frac{C(A_1, \bar{A}_1)}{D(A_1, \bar{A}_1)} = C(A_1, \bar{A}_1). \]

By Lemma 2.2, \((A, \bar{A})\) is a sparsest cut in \(G\). But \(D(B, \bar{B}) = 1\) for every uv separating cut \((B, \bar{B})\). Therefore, \((A_1, \bar{A}_1)\) is a minimum capacity separating uv cut. Q.E.D
Section 6.2 Duality of the MCFP

Consider an instance $G, D, C$ of the MCFP such that $G$ is isomorphic to $K_{2,3}$ and $D$ and $C$ are both unit functions. It is easy to see that $\hat{\lambda} = \frac{2}{3}$ and that the density of a sparsest cut is $\frac{1}{2}$. Therefore there are instances of the MCFP where the inequality in Lemma 2.2 is always strict for all cuts. Thus the sparsest cut problem is not the dual of the MCFP. Notice this is not unexpected, since the MCFP is solvable in polynomially bounded time and the sparsest cut problem is NP-hard. We now show the upper bound in Lemma 2.3 is always tight for a particular distance function, thus providing a duality theorem analogous to the max-flow min-cut theorem. We first prove a lemma which is helpful to show our main result.

Lemma 6.2. Let $G = (V, E)$ be a graph with a distance function $\hat{d}$. Define $\hat{d}$ as

$$\hat{d}(e) = \hat{d}(i, j) \quad \text{for all } e \in E \text{ with } e = ij.$$ 

Then

$$\hat{d}(k, l) = \hat{d}(k, i) \quad \text{for all } k, l \in V.$$

Proof. Let $k, l \in V$. Assume $p_1 : k = k_1, k_2, \ldots, k_n = l$, $n \geq 2$ is a path with $\hat{d}(p_1) = \hat{d}(k, l)$. Then

$$\hat{d}(k, l) = \sum_{e \in p_1} \hat{d}(e) = \sum_{i=1}^{n-1} \hat{d}(k_i, k_{i+1}) \geq \hat{d}(k, l).$$

Similarly, let $p_2 : k = j_1, j_2, \ldots, j_m = l$, $m \geq 2$ be a path with $\hat{d}(p_2) = \hat{d}(k, l)$. Then

$$\hat{d}(k, l) = \sum_{e \in p_2} \hat{d}(e) = \sum_{i=1}^{m-1} \hat{d}(j_i, j_{i+1}) = \sum_{e \in p_2} \hat{d}(e) = \hat{d}(p_2) \geq \hat{d}(k, l).$$

Thus

$$\hat{d}(k, l) = \hat{d}(k, l).$$

Q.E.D

Our main result is the following.
Theorem 6.3. Let $G, D, C$ represent an instance of the $MCFP$ with a maximum concurrent flow of throughput $\hat{Z}$ such that either $D$ or $C$ is a constant function. Let

$$\zeta(G, D, C) = \min_d \{ \frac{\sum_{e \in E} d(e) C(e)}{\sum_{i < j} d(i, j) D(i, j)} \}.$$ 

Then

$$\hat{Z} = \zeta(G, D, C),$$

(6.1)

where $d$ is a distance function on $G$.

Proof. The problem of finding $\zeta(G, D, C)$ can be expressed as a linear program, as in [Ma86]. Therefore $\zeta(G, D, C)$ exists. Now consider the following two cases:

Case 1. $C$ is a constant function, that is,

$$C(e) = c^* \text{ for all } e \in E.$$ 

Assume, to the contrary, that $\hat{Z} \neq \zeta(G, D, C)$. By Lemma 2.3,

$$\hat{Z} < \zeta(G, D, C).$$

(6.2)

Let $\epsilon = \frac{(G, D, C) - \hat{Z}}{2c^*} (\sum_{i < j} D(i, j))$, then $\epsilon > 0$. By Theorem 4.1 there exists a distance function $d$ such that

$$\hat{Z} \geq \frac{\sum_{e \in E} d(e) C(e)}{\sum_{i < j} d(i, j) D(i, j)} \zeta(G, D, C) - \frac{\hat{Z}}{2}$$

or

$$\frac{\hat{Z}}{2} \geq \frac{\sum_{e \in E} d(e) C(e)}{\sum_{i < j} d(i, j) D(i, j)} \zeta(G, D, C).$$

But $\frac{\sum_{e \in E} d(e) C(e)}{\sum_{i < j} d(i, j) D(i, j)} \geq \zeta(G, D, C)$, therefore $\frac{\hat{Z}}{2} \geq \frac{(G, D, C)}{2}$ or

$$\hat{Z} \geq \zeta(G, D, C).$$

However, this contradicts (6.2).
Case 2. \( D \) is a constant function, that is,

\[
D(i, j) = D_0 \quad \text{for all distinct } i, j \in V.
\]

We will use the extended capacity function \( C \) and Theorem 5.2 to construct an instance of the MCFP with uniform capacity. By Case 1, we have

\[
\bar{z} = \frac{\sum_{e \in E} \bar{d}(e)D_0}{\sum_{i < j} \bar{d}(i, j)(c^* - C(i, j))},
\]

where \( \bar{z} \) is the throughput of a maximum concurrent flow for the constructed instance, and \( \bar{d} \) is a distance function on \( G \). Notice that

\[
\bar{d}(e) = \bar{d}(i, j) \quad \text{for all } e \in E \text{ with } e = ij,
\]

for otherwise there exists an edge \( e = kl \) such that \( \bar{d}(e) > \bar{d}(k, l) \). Then for the distance function \( \bar{d} \) defined by

\[
\bar{d}(e) = \bar{d}(i, j) \quad \text{for all } e \in E \text{ with } e = ij,
\]

we have \( \sum_{e \in E} \bar{d}(e) < \sum_{e \in E} \bar{d}(e) \). Also, by Lemma 6.2, \( \bar{d}(i, j) = \bar{d}(i, j) \) for all \( i, j \in V \); thus

\[
\frac{D_0 \sum_{e \in E} \bar{d}(e)}{\sum_{i < j} \bar{d}(i, j)(c^* - C(i, j))} < \frac{D_0 \sum_{e \in E} \bar{d}(e)}{\sum_{i < j} \bar{d}(i, j)(c^* - C(i, j))},
\]

which is a contradiction. Therefore \( \bar{d} \) has the property indicated in (6.3). Hence

\[
\bar{z} = \frac{D_0 \sum_{i < j} \bar{d}(i, j)}{c^* \sum_{i < j} \bar{d}(i, j) - \sum_{i < j} \bar{d}(i, j)C(i, j)} = \frac{D_0 \sum_{i < j} \bar{d}(i, j)}{c^* \sum_{i < j} \bar{d}(i, j) - \sum_{e \in E} \bar{d}(e)C(e)}.
\]

(6.4)

Now observe that by Theorem 5.2, \( \bar{z} = \frac{c^*}{D_0} - \frac{1}{\bar{z}} \) or

\[
\bar{z} = \frac{D_0}{c^* - D_0 \bar{z}}.
\]

(6.5)

Substituting \( \bar{z} \) using (6.5) in (6.4) and solving for \( \bar{z} \), gives us

\[
\bar{z} = \frac{c^*}{D_0} - \frac{c^* \sum_{i < j} \bar{d}(i, j) - \sum_{e \in E} \bar{d}(e)C(e)}{\sum_{i < j} \bar{d}(i, j)D_0} = \]
\[
\frac{\sum_{e \in E} d(e)C(e)}{\sum_{i \neq j} d(i,j)D_0} = \frac{\sum_{e \in E} d(e)C(e)}{\sum_{i < j} d(i,j)D(i,j)}.
\]

This implies that (using Lemma 2.3)

\[
\zeta(G, D, C) = \frac{\sum_{e \in E} d(e)C(e)}{\sum_{i \neq j} d(i,j)D(i,j)}.
\]

Thus \( \hat{Z} = \zeta(G, D, C) \). \textbf{Q.E.D.}

As an immediate result of Theorems 6.1 and 6.3, we have the following.

**Theorem 6.4.** Let \( G, D, C \) be an instance of the \textit{MCFP} with two distinct vertices \( u, v \in V \) and a maximum concurrent flow of throughput \( \hat{Z} \) such that

\[
D(i,j) = \begin{cases} 
0 & \text{if } \{i,j\} \neq \{u,v\} \\
1 & \text{otherwise.}
\end{cases}
\]

and

\[
C(e) = c^* \text{ for all } e \in E.
\]

Then

\[
\hat{Z} = Min\{\lambda \{C(A, \bar{A})\} = Min\{\frac{\sum_{e \in E} d(e)C(e)}{\sum_{i < j} d(i,j)D(i,j)}\},
\]

where \((A, \bar{A})\) is a uv separating cut and \( d \) is a distance function.

Theorem 6.4 indicates that if the capacities on the edges are the same, then the undirected version of the max flow min-cut theorem is a special case of the duality theorem, that is, the duality theorem is more general than the max-flow min-cut theorem.

The general version of the duality theorem where demand and capacity are both arbitrary functions can be shown using the duality theory of linear programming. However, this technique does not explore the relationship between the structure of the dual problem and the original problem, as our flow rerouting algorithm does.
CHAPTER VII

APPLICATIONS OF THE MCFP

In this chapter we discuss applications of the \textit{MCFP} to the routing problem and cluster analysis.

Section 7.1 The Routing Problem in Telecommunications

A distributed network is a network of host computers that provides services to the users. The hosts are connected by the communication subnet which carries the host to host messages. The subnet itself consists of two basic components: switching elements which are specialized computers and transmission lines. Following the ARPANET terminology, we will call the switching elements IMPs (Interface Message Processors) and the transmission lines channels. Given an initial topology for the network (which is subject to change), we are concerned with the question of if and how the network can concurrently support all the demands for the traffic message flow between hosts at that instant, subject to the operational capacity of the channels (routing problem). In general, the routing problem is to determine a set of routes and the messages which are sent on them such that a well defined objective function is optimized (minimize total delay or cost, maximize the throughput, etc.). Our particular concern in solving the routing problem is how to efficiently and adaptively compute optimal message flow routings to maximize the throughput that can be maintained over time in a dynamic environment allowing changes in the distribution of traffic demands, the operational capacity of the channels, and the topological and/or hierarchical structure of the network. The classical model for packet switched communication networks employs a fixed network topology and fixed channel capacities for the duration of the time frame.

43
Within this frame, the traffic requirements are allowed to vary to the extent that each origin-destination flow requirement is governed by an independent Poisson process. This formulation yields a rather formidable multicommodity flow problem [KH80] with a nonlinear objective function which does not solve networks with more than 30 nodes. Computational success on moderate size problems is achievable by procedures such as the traditional flow deviation [FC71]. However, these approaches do not appear to identify the underlying structure of the problem in anywhere near the clarity as for the theory of the maximum concurrent flow. Further perturbations to this model to account for changes over time of the network topology or variable traffic requirements other than by a Poisson process (e.g., bursts or shifts in average usage) only lead further away from understanding the structure of the underlying problem. The more complex time varying environment resulting from this model then seems to require analysis by simulation without benefits from understanding the implicit structure of the underlying static problem.

We believe a sound approach to understanding and resolving questions about routing in a dynamic environment is to understand first the nature and the structure of an optimal solution for maximising concurrent flow in a static network with (relative) levels of flow requirements between all origin-destination pairs.

Section 7.2 Applications of the MCFP to the Routing Problem

For a given incremental time frame we assume that we have a distributed network with fixed capacities on the channels and fixed levels of flow to be supported concurrently between all origin destination pairs in the network. Our measure of throughput is specified by a parameter $Z$ such that we assume we can handle flow equal to $Z$ times the requirement between each origin destination concurrently. Our objective is to find the peak (maximum) throughput $\hat{Z}$ that can be hosted subject to the capacity constraints. Notice that then, the routing problem and
the MCFP are essentially the same problems if we assume a zero demand for all IMP to IMP and host to IMP communications. The rich structure of the MCFP provides a full understanding of the underlying static routing problem. For example, Theorem 2.5 indicates that the channels that must be saturated to capacity under any optimal routing policy constitute a $k$ partite cut, that is, a partition of IMPs and hosts into $k$ parts where $k$ may be greater than 2. This helps to understand and analyze the behavior of the network and the bottlenecks during rush hours. The efficient iterative flow routing algorithms that we presented in Chapters 3 and 4 can be used to solve the routing problem in a static environment. The desirable properties of a good routing policy as indicated in [TA81] are: correctness, efficiency, robustness, stability, fairness, and optimality. Empirical results in Chapter 3 indicate that our routing algorithms have all these properties. Consider that if we are operating with throughput say 60 percent of peak level, then the concurrent flow can be routed by our specified routing table so as to utilize only 60 percent of the capacity of each channel. Such a scheduled flow provides maximum uniform capability to adapt to changes in the network flow demand and/or topology. Also, the iterative nature of our flow rerouting algorithm facilitates adaptive incorporation of the changing demands. The efficient nature of our algorithms (as far as the running time and the storage are concerned) also indicate their flexibility to be adjusted to changes in capacities, topology, demands, etc.

Section 7.3 Applications of Duality Theorem to the Routing Problem

Studies of the routing in [FC71] employ a stochastic model with constraints similar to that of the MCFP. The objective in this model is to minimize a non-linear objective function representing the total delay in the network (which is very similar to the distance ratio in the duality theorem), given stochastically determined delays on each channel. If we assume $d(e)$ in (6.1) represents the delay time per unit capacity on the edge $e$, then $\sum_{i,j} d(i,j) D(i,j)$
is the total delay that all the source destination pairs are experiencing. Note that

$$\frac{\sum_{e \in E} d(e) C(e)}{\sum_{i < j} d(i,j) D(i,j)}$$

denotes what percentage of the total delay for all source-destination pairs is introduced by the delay on the channels (edges). The smaller this percentage, the larger is the total delay, introduced in the network corresponding to a fixed amount of sum of delays on all channels. The duality theorem indicates that the worse case delay experienced by users of the network will happen when the throughput is maximized. Furthermore, (6.1) analytically shows the correspondence between the delay experienced by the network and the optimal throughput, indicating what delay the users expect to see when the throughput has its maximum value.

Section 7.4 Applications of the MCFP to Cluster Analysis

A dissimilarity (similarity) over a set $S$ is a function $l: S \times S \rightarrow \mathbb{R}^+$ such that $l(i,j) = l(j,i)$ and $l(i,i) = 0$ for all $i, j \in S$. Let $S_1, S_2 \subseteq S$. Then the average dissimilarity (similarity) between $S_1$ and $S_2$ is

$$\frac{\sum_{i \in S_1, j \in S_2} l(i,j)}{|S_1||S_2|}.$$

(7.1)

Let $S$ be a set of objects with a dissimilarity function $l$. A cluster in $S$ is a maximal collection of suitably similar objects. The cluster analysis techniques seek to find a partition of objects into clusters, or in general, to determine a hierarchy of the cluster partitions. This hierarchy is represented using a tree or dendogram. A bottom up clustering method would form a larger cluster from the two closest smaller clusters, whereas a top down (divisive) technique would partition a larger cluster into the two most distant smaller clusters. For a detailed discussion of clustering techniques, see [SS73], [GO81], and [Ma86]. A bottom up clustering technique UPGMA (unweighted pair group method based on arithmetic average) is widely used and is described below as an algorithm.
Algorithm Cluster1

/ The input is the dissimilarity (similarity) function $l$ over a set $S$. /

/ The output is a hierarchy of clusters. /

Start with $|S|$ clusters such that every object of $S$ is in one cluster by itself.

WHILE there is more than one cluster DO

Form a new cluster from the two closest clusters, where the measure of the closeness is the average dissimilarity (similarity) between the clusters.

Update the dissimilarities (similarities) between the clusters using (7.1).

END

It is straightforward that the order of time complexity for this algorithm is $O(|S|^3)$. A more efficient UPGMA clustering algorithm of time complexity $O(|S|^2)$ is given in [MU83]. Although UPGMA can be computed efficiently, it suffers from the drawback of correct grouping of small clusters at the expense of possible errors in grouping large clusters [MA86], [TM86]. We now discuss the top down clustering technique DPGMA (divisive pair group method based on arithmetic average) which is introduced in [MA86]. This technique is best described by the recursive procedure Cluster2 given below.

Procedure Cluster2 (A)

/ The input is the dissimilarity (similarity) $l$ on $S$. The output is a hierarchy of the clusters constructed top down from cluster A. /

IF Cluster A contains more than one object

THEN Partition A into two clusters $A_1$, $A_2$ such that the average dissimilarity (similarity) between $A_1$ and $A_2$, defined by (6.1) is maximised (minimised) over all partitions of A into two clusters.

Cluster2($A_1$)

Cluster2($A_2$)
Cluster2(S) terminates after at most |S| recursive calls, since every time it invokes itself, the number of clusters is increased by 1. DPGMA and UPGMA dendograms are illustrated in the appendix (Figure 5). The problem of partitioning a cluster into two smaller clusters such that the average dissimilarity (similarity) between the new clusters is maximized (minimized) is equivalent to the densest cut (sparsest cut) problem. Therefore DPGMA is NP-hard. However, in Chapter 3 the applications of the MCFP in solving the sparsest cut problem were mentioned. Thus the MCFP is of potential use to the DPGMA. We now formalize the relationship of the MCFP to the DPGMA.

Let V be the set of objects to be clustered with the dissimilarity (similarity) function I. Define an instance of the MCFP with uniform capacity (uniform demand) such that 

\[ G = (V, E) \]

is a complete graph and \( D = I \) (\( C = I \)). Then

\[ 2 \leq \frac{C(B,B)}{D(B,B)}, \]

where (\( B, B \)) is a sparsest cut in G. If I is a similarity function, then we have

\[ C(B,B) = \min \left\{ \frac{\sum_{i \in A, j \in B} I(i,j)}{|A||B|} \right\}. \]

Now assume that I is a dissimilarity function. Then

\[ \frac{C(B,B)}{D(B,B)} = \frac{|B||B|}{\sum_{i \in B, j \in B} I(i,j)} = \min \left\{ \frac{|A||A|}{\sum_{i \in A, j \in A} I(i,j)} \right\}, \]

implying that

\[ \frac{\sum_{i \in B, j \in B} I(i,j)}{|B||B|} = \max \left\{ \frac{\sum_{i \in A, j \in A} I(i,j)}{|A||A|} \right\}. \]

Therefore the sparsest cut (\( B, B \)) identifies a partition \( \{B, B\} \) with the maximum average similarity (minimum average similarity) as desired in DPGMA. If for some instances of the MCFP
the inequality (7.2) is sharp, then a sparsest cut or at least a sparse cut may be identified using the cut constructed in step 2 of Algorithm $R$. This cut identifies the appropriate partitioning of $V$ for the DPGMA. Recall the empirical results of Chapter 2 which are summerized in Table 1. Consider all cases with no *, the presence of an edge is interpreted as a similarity of 1 for the end vertices of the edge, while the absence of an edge is viewed as a similarity of 0 for the corresponding vertices. Theorem 3.3 indicates that in all such cases the MCFP identifies a sparsest cut $(B, \overline{B})$, implying that $\{B, \overline{B}\}$ is the desired partitioning of $V$ into two smaller clusters. This process can be repeated for each new cluster to obtain the whole hierarchy or the dendogram.

In [TM86] such a technique has been utilized to implement the DPGMA. This implementation is slow and does not solve the problems beyond 35 objects. In many instances of the clustering problem the dissimilarities between the objects are Euclidean distances. Therefore, it is of practical interest to investigate the applications of the MCFP to this particular class of the clustering problems. Table 2 in the appendix summarizes the empirical results regarding instances of the MCFP, where the demand is an Euclidean distance function in $E^n$, $n = 1, 2$. The cuts identified by the algorithm in 2 cases are shown in the appendix. Surprisingly, in all cases the cut identified by our algorithm happens to be a sparsest cut. Furthermore, in all cases the cut is identified with a separating hyperplane, that is, if $(B, \overline{B})$ denotes the sparsest cut found by the algorithm, then the vertices in $B$ are separated from the vertices in $\overline{B}$ by a hyperplane.

This suggests the following conjecture.

**Conjecture.** The sparsest cut problem for the MCFP with uniform capacity (DPGMA clustering) problem is solvable in polynomially bounded time if the demands between vertices (dissimilarities between pairs) are Euclidean distances in $E^n$, $n \geq 1$. Furthermore, there exists at least one sparsest cut that is identified with a separating hyperplane. Note that the identification of a sparsest cut with a separating hyperplane implies the existence of polynomially bounded algorithms which can directly solve the sparsest cut problem.
Table 1

<table>
<thead>
<tr>
<th>Graph</th>
<th>Num. of vertices</th>
<th>Num. of edges</th>
<th>Avg. num. of active paths</th>
<th>Avg. length of active paths</th>
<th>Cpu time (sec)</th>
<th>Z</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>7</td>
<td>1.16</td>
<td>1.70</td>
<td>2.8</td>
<td>.4993</td>
<td>.5000</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>14</td>
<td>1.11</td>
<td>1.62</td>
<td>0.5</td>
<td>.1999</td>
<td>.2000</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>15</td>
<td>1.50</td>
<td>1.88</td>
<td>2.3</td>
<td>.3120</td>
<td>.3125</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>26</td>
<td>1.49</td>
<td>2.77</td>
<td>11.7</td>
<td>.0799</td>
<td>.0800</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>31</td>
<td>1.50</td>
<td>1.79</td>
<td>7.0</td>
<td>.2590</td>
<td>.2592</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>49</td>
<td>1.66</td>
<td>1.80</td>
<td>21.7</td>
<td>.2776</td>
<td>.2777</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>33</td>
<td>1.25</td>
<td>3.80</td>
<td>13.4</td>
<td>.0290</td>
<td>.0300</td>
</tr>
<tr>
<td>8</td>
<td>31</td>
<td>188</td>
<td>1.88</td>
<td>2.45</td>
<td>67.0</td>
<td>.1997</td>
<td>.2000</td>
</tr>
<tr>
<td>9,6</td>
<td>10</td>
<td>24</td>
<td>1.87</td>
<td>1.72</td>
<td>4.5</td>
<td>.3635</td>
<td>.3636</td>
</tr>
<tr>
<td>9,12</td>
<td>20</td>
<td>96</td>
<td>2.17</td>
<td>1.77</td>
<td>23.9</td>
<td>.3379</td>
<td>.3380</td>
</tr>
<tr>
<td>16,24</td>
<td>40</td>
<td>384</td>
<td>2.39</td>
<td>1.79</td>
<td>96.0</td>
<td>.3245</td>
<td>.3265</td>
</tr>
<tr>
<td>32,48</td>
<td>80</td>
<td>1536</td>
<td>2.17</td>
<td>1.70</td>
<td>819.6</td>
<td>.3209</td>
<td>.3210</td>
</tr>
<tr>
<td>50,50</td>
<td>100</td>
<td>2500</td>
<td>2.16</td>
<td>1.77</td>
<td>1490.0</td>
<td>.3376</td>
<td>.3378</td>
</tr>
<tr>
<td>Q_3</td>
<td>8</td>
<td>12</td>
<td>1.58</td>
<td>1.94</td>
<td>2.3</td>
<td>.2499</td>
<td>.2500</td>
</tr>
<tr>
<td>Q_b</td>
<td>16</td>
<td>32</td>
<td>1.53</td>
<td>2.37</td>
<td>9.2</td>
<td>.1299</td>
<td>.1250</td>
</tr>
<tr>
<td>Q_5</td>
<td>32</td>
<td>80</td>
<td>1.54</td>
<td>2.16</td>
<td>36.3</td>
<td>.6249</td>
<td>.6250</td>
</tr>
<tr>
<td>Q_b</td>
<td>64</td>
<td>192</td>
<td>1.32</td>
<td>3.13</td>
<td>148.0</td>
<td>.3124</td>
<td>.3125</td>
</tr>
<tr>
<td>random</td>
<td>60</td>
<td>895</td>
<td>2.10</td>
<td>1.75</td>
<td>334.0</td>
<td>.2840</td>
<td>.2844</td>
</tr>
<tr>
<td>random</td>
<td>100</td>
<td>2446</td>
<td>2.13</td>
<td>1.77</td>
<td>805.8</td>
<td>.3218</td>
<td>.3261</td>
</tr>
<tr>
<td>random</td>
<td>120</td>
<td>3598</td>
<td>2.23</td>
<td>1.77</td>
<td>1296.0</td>
<td>.3345</td>
<td>.3364</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table 2

Results of Application of Algorithm R to Graphs 21-33

<table>
<thead>
<tr>
<th>Graph no.</th>
<th>Num. of vertices</th>
<th>Num. of edges</th>
<th>Avg. num. of paths per pair</th>
<th>Avg. path length</th>
<th>Cpu time</th>
<th>Z</th>
<th>Upper bound</th>
<th>Num. of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>10</td>
<td>45</td>
<td>2.4</td>
<td>1.9</td>
<td>9</td>
<td>.00216</td>
<td>.00217</td>
<td>9</td>
</tr>
<tr>
<td>22</td>
<td>15</td>
<td>105</td>
<td>2.3</td>
<td>1.9</td>
<td>23</td>
<td>.00137</td>
<td>.00137</td>
<td>14</td>
</tr>
<tr>
<td>23</td>
<td>20</td>
<td>190</td>
<td>2.2</td>
<td>2.1</td>
<td>42</td>
<td>.00170</td>
<td>.00170</td>
<td>14</td>
</tr>
<tr>
<td>24</td>
<td>30</td>
<td>435</td>
<td>2.2</td>
<td>2.2</td>
<td>179</td>
<td>.00197</td>
<td>.00197</td>
<td>19</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>45</td>
<td>2.3</td>
<td>2.4</td>
<td>7.8</td>
<td>.01363</td>
<td>.01364</td>
<td>8</td>
</tr>
<tr>
<td>26</td>
<td>15</td>
<td>105</td>
<td>2.1</td>
<td>1.86</td>
<td>13.1</td>
<td>.04051</td>
<td>.04058</td>
<td>7</td>
</tr>
<tr>
<td>27</td>
<td>20</td>
<td>190</td>
<td>1.9</td>
<td>1.92</td>
<td>32.7</td>
<td>.01251</td>
<td>.01251</td>
<td>9</td>
</tr>
<tr>
<td>28</td>
<td>25</td>
<td>300</td>
<td>2.65</td>
<td>2.64</td>
<td>130</td>
<td>.01365</td>
<td>.01367</td>
<td>20</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>435</td>
<td>2.07</td>
<td>1.92</td>
<td>125</td>
<td>.01321</td>
<td>.01325</td>
<td>13</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>780</td>
<td>1.67</td>
<td>1.96</td>
<td>271</td>
<td>.01386</td>
<td>.01388</td>
<td>14</td>
</tr>
<tr>
<td>31</td>
<td>50</td>
<td>1225</td>
<td>1.88</td>
<td>1.9</td>
<td>477</td>
<td>.013833</td>
<td>.01385</td>
<td>18</td>
</tr>
<tr>
<td>32</td>
<td>60</td>
<td>1770</td>
<td>1.7</td>
<td>1.9</td>
<td>721</td>
<td>.00459</td>
<td>.00459</td>
<td>17</td>
</tr>
<tr>
<td>33</td>
<td>80</td>
<td>3160</td>
<td>1.93</td>
<td>1.86</td>
<td>1850</td>
<td>.00278</td>
<td>.00278</td>
<td>23</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table 3

Results of Application of Algorithm R with Different Tolerance Values

<table>
<thead>
<tr>
<th>GRAPH</th>
<th>$\epsilon = 0.002$</th>
<th>$\epsilon = 0.01$</th>
<th>$\epsilon = 0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU TIME</td>
<td>NUM. ITERATIONS</td>
<td>CPU TIME</td>
</tr>
<tr>
<td>1</td>
<td>2.800</td>
<td>4</td>
<td>2.000</td>
</tr>
<tr>
<td>2</td>
<td>.5160</td>
<td>1</td>
<td>.5200</td>
</tr>
<tr>
<td>3</td>
<td>2.30</td>
<td>5</td>
<td>1.975</td>
</tr>
<tr>
<td>4</td>
<td>11.72</td>
<td>7</td>
<td>7.833</td>
</tr>
<tr>
<td>5</td>
<td>7.070</td>
<td>7</td>
<td>7.001</td>
</tr>
<tr>
<td>6</td>
<td>21.70</td>
<td>13</td>
<td>15.01</td>
</tr>
<tr>
<td>7</td>
<td>13.44</td>
<td>5</td>
<td>10.40</td>
</tr>
<tr>
<td>8</td>
<td>67.00</td>
<td>19</td>
<td>59.95</td>
</tr>
<tr>
<td>K4,6</td>
<td>4.600</td>
<td>6</td>
<td>4.100</td>
</tr>
<tr>
<td>K6,12</td>
<td>23.90</td>
<td>8</td>
<td>18.02</td>
</tr>
<tr>
<td>K16,24</td>
<td>96.00</td>
<td>11</td>
<td>90.00</td>
</tr>
<tr>
<td>K32,48</td>
<td>819.6</td>
<td>14</td>
<td>699.6</td>
</tr>
<tr>
<td>K50,50</td>
<td>1490.0</td>
<td>15</td>
<td>1372.5</td>
</tr>
<tr>
<td>Q3</td>
<td>2.300</td>
<td>4</td>
<td>1.683</td>
</tr>
<tr>
<td>Q4</td>
<td>9.200</td>
<td>5</td>
<td>9.220</td>
</tr>
<tr>
<td>Q5</td>
<td>36.30</td>
<td>5</td>
<td>26.71</td>
</tr>
<tr>
<td>Q6</td>
<td>148.0</td>
<td>5</td>
<td>120.3</td>
</tr>
<tr>
<td>RANDOM</td>
<td>334.0</td>
<td>10</td>
<td>179.1</td>
</tr>
<tr>
<td>RANDOM</td>
<td>665.8</td>
<td>8</td>
<td>703.8</td>
</tr>
<tr>
<td>RANDOM</td>
<td>1296.0</td>
<td>10</td>
<td>1000.3</td>
</tr>
</tbody>
</table>
Figure 2. Graphs 1-8.
Figure 3. The Sparsest Cut for Graph 25.
Figure 4. The Sparsest Cut for Graph 28.
Figure 5. UPGMA and DPGMA dendograms [Ma86]. Each adjacent pair has a dissimilarity 1 and each non adjacent pair has a dissimilarity 10.


