MINIMUM CONCAVE COST MULTICOMMODITY NETWORK DESIGN

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FATİH SAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Cüneyt F. BAZLAMAÇCI
Supervisor

Examining Committee Members

Prof. Dr. Hasan GÜRAN (METU, EE)	
Asst. Prof. Dr. Cüneyt F. BAZLAMAÇCI (METU, EE)	
Prof. Dr. Semih BİLGEN (METU, EE)	
Dr. Ece Ş. (GÜRAN) SCHMIDT (METU, EE)	
Erkan Ünal, M.Sc. (PETSOFTEL)	

ABSTRACT

MINIMUM CONCAVE COST MULTICOMMODITY NETWORK DESIGN

SAY, Fatih
M.Sc., Department of Electrical and Electronics Engineering
Supervisor : Asst. Prof. Dr. Cüneyt F. BAZLAMAÇCI

September 2005, 90 pages

Minimum Concave Cost Multicommodity Network Design Problem arises in many application areas, such as transportation planning, distributed energy system and especially both circuit and packet switching backbone network design. Exact concave optimization algorithms have been developed, but these methods are applicable if the network size is small. Therefore, these problems are usually solved by non-exact iterative methods.

In this thesis work, methods proposed for circuit switching and packet switching network design are evaluated in detail. After a comprehensive literate survey, Yaged's Linearization, Minoux greedy and Minoux accelerated greedy methods are found to be applicable to circuit switching network design when both solution quality and computational time is considered. Previously, it has been found that Minoux greedy methods may create routings with cycles and in order to eliminate these cycles a modification has been proposed. In this work, this modification is extended and evaluated in detail. Similarly, Gerla and Kleinrock's Concave Branch

Elimination, Gersht's greedy and Stacey's Concave Link Elimination methods are investigated within the context of packet switching network design.

All of these methods consider aggregate flows on each link simultaneously re-routing more than one commodity in one step. This thesis work also considers an alternative disaggregate approach, where only one commodity is handled at a time.

Finally, algorithms proposed for circuit switching network design problem are adapted to the packet switching case and an extensive comparative computational study is performed to point out the best method with respect to time and solution quality for a number of networks and cost structure. Computational results have shown that modification on Minoux greedy to eliminate cycles leads to considerable improvements and the disaggregate approach gives the best result in some networks and cost structure.

Keywords: Concave Cost Network Design, Multicommodity Flow Problem, Disaggregate Local Search.

ÖZ

ÇOK ÜRÜNLÜ EN KÜÇÜK İÇBÜKEY MALİYETLİ AĞ TASARIMI

SAY, Fatih
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği
Tez Yöneticisi : Asst. Prof. Dr. Cüneyt F. BAZLAMAÇCI

Eylül 2005, 90 sayfa

Çok ürünlü içbükey maliyetli en küçük ağ tasarımının başta devre ve paket anahtarlamalı omurga ağ tasarımı olmak üzere; taşımacılık planlaması, enerji dağıtım şebekeleri gibi bir çok alanda uygulamaları vardır. Bu problemi optimum çözen yöntemler geliştirilmiş olmasına rağmen, işlem süresi düşünüldüğünde bu yöntemler sadece küçük ölçekli ağlara uygulanabilmektedir. Bu nedenle, bu tür problemler kesin olmayan tekrarlama yöntemleriyle çözülmektedir.

Bu çalışma kapsamında devre ve paket anahtarlamalı ağ tasarımı için önerilen yöntemler detaylı olarak incelenmektedir. Yapılan kapsamlı literatür araştırması sonucunda Yaged tarafından önerilen doğrusallaştırma metodu, Minoux'un önerdiği bağlantı azalma ve yine Minoux'un önerdiği hızlandırılmış bağlantı azaltma yöntemlerinin devre anahtarlamalı ağ tasarımlarına uygun olduğu görülmüştür. Minoux'un önerdiği yöntemlerde döngü içeren rotalar oluşabileceği önceki çalışmalarda gözlenmiştir. Bu çalışma kapsamında, Minoux yöntemleri için önerilen değişiklikler detaylarıyla incelenmiş ve geliştirilmiştir. Benzer şekilde, yapılan araştırmalar sonucunda, Gerla ve Kleinrock tarafından önerilen içbükey kol eleme, Gersht'in önerdiği bağlantı azaltma ve Stacey'in önerdiği içbükey bağlantı eleme metotları paket anahtarlamalı ağ tasarıma uygun yöntemler olarak bulunmuş ve detaylı olarak değerlendirilmiştir.

İncelenen tüm yöntemler, tek adımda her bağlantı üzerindeki toplam trafiği yeniden yönlendirmektedir. Bu çalışmada, her aşamada sadece bir ürünün(trafiğin) ele alındığı alternatif bir ayrıştırma yöntemi de değerlendirilmiştir. Ayrıca devre anahtarlamalı ağ tasarımı için önerilen yöntemler paket anahtarlı ağ tasarımlarına uyarlanmıştır. Yöntemlerin çözüm kalitesini incelemek ve çeşitli maliyet yapıları ve ağ tipleri için hangi yöntemin en iyi çözümü ürettiğini görmek için kapsamlı bir hesaplama çalışması yapılmıştır. Çalışma sonuçlarında, Minoux metoduna yapılan döngü yok etme değişikliklerinin çözüm kalitesinde iyileştirmeler sağladığı ve alternatif ayrıştırma yaklaşımının çeşitli ağ tiplerinde en iyi çözümü sağladığı görülmüştür.

Anahtar Kelimeler: İçbükey maliyetli ağ tasarımı, Çok ürünlü akış problemi, Ayrıştırılmış yerel arama yöntemi.

To my wife

&

My little son

For their Love & Support

ACKNOWLEDGEMENTS

I would like to express my thanks to my supervisor Asst. Prof. Dr. Cüneyt F. Bazlamaçcı for his guidance and support during the preparation of this study.

I wish to express my thanks to my colleagues and Aselsan for computational resource support during this study.

Finally, I wish to express my gratitude to my wife Şifa SAY for her precious support, assistance and unattainable understanding at every stage of this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

The problem of minimum cost network design in order to satisfy a predefined set of system requirements (such as traffic demands, timing constraints, reliability issues, desired level of security, quality of service and so on) while minimizing the total network cost, arises in many application areas, such as, transportation planning, plant location and capacity expansion, production planning, waste water resource management, distributed systems, energy systems and especially in computer and telecommunication networks, providing a variety of services, such as voice, data and video to subscribers.

The network under consideration in this work is a backbone communication network, or a computer network. However, this work can easily be extended to the other network types, like transshipment or energy distribution systems.

The marginal cost of links in these backbone networks decreases as the link capacity increases, i.e. these links show strong economies of scale. Therefore, concave cost functions are used to model the link costs. The traffic requirements in these backbone networks, requires a number of different source to destination pairs hence the network is multicommodity network. Therefore, the problem discussed in this work is called Minimum Concave Cost Multicommodity Network Design Problem (MCMNDP).

In addition to backbone networks, the MCMNDP is applicable to private networks, which are based on leased lines. As the leased lines are billed on a fixed rate which is independent of usage and they could be reconfigured to accommodate changes in traffic patterns and price and to provide improved reliability, it is possible to produce significant cost benefits by carefully approximating the cost using analytical concave cost functions and using effective algorithms.

The minimum cost network design process involves, the configuration of the backbone network, finding the route of network traffic and finding the link capacities subject to certain performance requirements. This task is enormous and there is no such thing as a closed-form algorithm for optimizing a network topology for most applications [22]. In conventional approaches, each component of the network is treated as a separate optimization task. However, it is not possible to optimize the network cost simply by overlying optimized sub-networks. Multicommodity flow problem with linear link costs in addition to fixed link installation costs, called as the fixed charge problem in the literature, has been studied earlier in [2,17] and shown to be an NP-hard problem. MCMNDP is a generalization of the uncapacitated fixed charge network design problem; therefore it is NP-hard, too. As a result, heuristic methods are used to solve MCMNDP.

The transmission media, like terrestrial microwave radio, communication satellite, optical fiber, in current telecommunications industry show large economies of scale, when their large bandwidth is well utilized. In the area of network design, discrete link costs are usually approximated by a continuous function to simplify the design problem. Although linear and offset-linear (fixed charge) cost functions simplify the problem, and are commonly used, they cannot model the economies of scale that often exist in the pricing of links. In this work concave link cost function is used to capture the effect of economies of scale in link capacity pricing.

In order to make use of economies of scale, different traffic requirements, such as voice, data and video, are combined. In this way, excess bandwidth becomes available with a small marginal cost. This thesis work is concerned with providing a minimum cost solution to the topological network design problem arising in the backbone design phase, with link costs functions modeled with general concave functions.

Backbone communication network models involving concave-cost functions have been studied by many researchers. Among them, Yaged [25, 26], Zadeh [27], Minoux [19, 20], Gerla and Kleinrock [11], Gersht and Weihmayer [12] and Stacey, Eyers and Anido [23] have proposed methods applicable to MCMNDP. All these methods are heuristic of the add/delete type, based on routing neighborhood properties. Due to the inherent difficulty imposed by concave costs, the models studied do not take into account any side constraints or capacities. They are also static models. However, they are still useful in the design of backbone networks within an iterative and hierarchical design framework.

In view of the potentially large sizes of the networks involved, general MCMNDP are solvable only by non-exact, approximate algorithms. Existing such algorithms fall into two categories: those which try to locate a local optimum and those which solve a series of approximations to the original problem, terminating when some nearness criterion is satisfied. The six methods which fall into the first category, namely Yaged’s linearization [25] and Minoux’s greedy [19,20] , Minoux’s accelerated greedy [19,20], Gerla and Kleinrock’s Concave Branch Elimination [11], Gersht and Weihmayer’s greedy [12] and Stacey, Eyer and Anido’s Concave Link Elimination [23] methods are not fully investigated, although the Yaged’s and Gerla and Kleinrock’s methods are widely referred. The former three methods are proposed for circuit switching networks while the latter three algorithms for packet switching networks. All these algorithms use aggregate (or total) flow and consider re-routing more than one origin-destination flow (commodity) when an edge is to be deleted. The algorithm developed in this work, namely Disaggregate Local Search, belongs to the same category; however, it proposes a local search technique based on re-routing of only a single commodity at each iteration. Though simple, this approach proved to be highly effective and it is conjectured that the local optimality conditions defined for the proposed local search are stronger than those on which existing methods are based.

In the present work, an improvement to the Minoux greedy algorithms, which leads to solutions of higher quality, is also evaluated. These improvements are based on the observation that these algorithms can get trapped in local optima which have unnecessary excess flow capacities. It is demonstrated that these excess capacities cannot be removed completely by a previously known strategy proposed by Minoux.

It has been realized that, the methods proposed for circuit switching MCMNDP can also be adapted to packet switching MCMNDP with some adaptation techniques. Gerla and Kleinrock's capacity assignment and Dutta and Lim's equal link utilization methods are evaluated as adaptation technique.

The rest of the thesis is organized as follows. First a formal definition of MCMCNDP is given for both circuit and packet switching networks. Then a literature survey in the backbone network design and concave cost flow area is given. In chapter 4, existing methods which are applicable to MCMNDP and in chapter 5, the modification to Minoux greedy algorithms are reviewed. Chapter 6 describes the Disaggregate Local Search method. Then the adaptation techniques and adaptation of circuit switching methods to packet switching MCMNDP is given in Chapter 7. Finally an extensive empirical computational study carried out to evaluate the performance of the methods in chapter 8. Chapter 9 concludes the thesis.

CHAPTER 2

PROBLEM DEFINITION

In the literature, two types of backbone networks are studied within the context of MCMNDP. In the first type, the network is composed of circuit switching links and there are no performance, fault tolerance and redundancy constraints. In the second type, packet switching links are used for the backbone network and some performance and reliability constraints must be met. Below, the MCMNDP associated with circuit switching and packet switching links are described in detail.

2.1 Minimum Concave Cost Multicommodity Circuit Switching Network Design Problem

This type of network design problems are encountered in transportation planning, waste water networks, energy distribution, private networks and backbone communication networks consisting of circuit switching links, such as telephony networks. In practice, there is an upper limit for the channel capacity. However, in this model, for the sake of simplicity the links are assumed to have no upper bound on the link capacity.

Minimum cost multicommodity network design problem for circuit switching networks can be described as follows. Given an undirected graph *G=[X,U]* consisting of a set X of n nodes and a set U of m links, coupled with n-vectors $b^k = \left(b_i^k\right)$(demand vector) and constant r^k (demand value) for every $k \in K$, where

K is the set of commodities. Let ψ_u be the aggregate flow on link *u* and a concave-cost function $\Phi_u(\psi_u)$ for each link *u*, solve

$$P = \left\{ \begin{array}{lll} \min \Phi(\psi) = \sum_{u \in U} \Phi_u(\psi_u) & & \\ \text{subject to} & & \\ \sum_{\{j \ni (i,j) \in X\}} \varphi_{ij}^k - \sum_{\{j \ni (j,i) \in X\}} \varphi_{ji}^k = \left\{ \begin{array}{ll} r^k & \text{if } i = s(k) \\ -r^k & \text{if } j = t(k) \\ 0 & \text{otherwise} \end{array} \right\} & & (1) \\ \psi_u = \sum_k \left| \varphi_u^k \right| & \forall u \in U & (2) \\ 0 \le \psi_u & \forall u \in U & (3) \end{array} \right\}$$

assuming all demands are integral. φ_u^k indicates the *k-th* individual flow component on link *u* and. Constraint (1) is the standard node conservation of flow for each commodity *k.* Multicommodity flow requirements are in fact a list of source sink pairs $s(k), t(k)\ \forall k \in K$ with corresponding prescribed flow values r^k, each r^k representing, for instance the traffic quantity which should be sent between *s(k)* and *t(k).*

This general model is more suitable for communication network synthesis although it might represent other problems equally well. Each commodity represents distinct traffic requirement or the same traffic requirement with different points of origin and destination (for example, a message or channel from Ankara to Istanbul is a commodity distinct from a message or channel from Ankara to Antalya).

Each concave-cost Φ_u is a non-decreasing and continuous function on [0, +∞) of the aggregate flow ψ_u on link *u*. In real problems, cost function are often closely approximated by analytic formulae of the form $\Phi_u(\psi_u) = F_u + l_u(\psi_u)^{\alpha_u}$ where F_u represent the fixed installation cost and $l_u > 0$. α_u represents the link concavity value and $0 < \alpha_u < 1$ [20].

The directed version of the MCMNDP was studied in detail in [3]. The major difference between the two is the underlying network. When bidirectional links are involved, the model is algebraic, that is, flows can be positive or negative, depending on the arbitrarily chosen direction convention. Therefore, the objective to be minimized is a function of the sum of absolute values of opposite flows on each link. In the context of telecommunication network design, the number of commodities is potentially high.

2.2 Minimum Concave Cost Multicommodity Packet Switching Network Design Problem

Minimum Concave Cost Multi Commodity Packet Switching Network Design Problems arise again in backbone telecommunication and computer network design. Due to the nature of the packet switching techniques, an additional performance requirement exists in such networks.

The design of packet switching networks is different and more complex than the design of circuit switching networks. The use of packet switching techniques requires the analysis of the relationship between packet delay, line utilization and buffer utilization.

In packet switching networks, a maximum packet delay from source to destination is generally required as the performance criterion. If the network topology and routing is known and the network traffic is purely deterministic, it is possible to satisfy the maximum packet delay requirement between each source and destination. However, when the network is designed based on the expected traffic requirement, it is difficult to satisfy the delay constraint between each source and destination pair. In this network design problem, average packet delay of the network is used as the performance criterion, instead.

The network to be designed is modeled as an undirected network $G = (X, U)$, where X is the set of n network nodes and U is the set of m links. Capacity c_{ij} is assigned to each link $(i, j) \in U$ to allow it to carry traffic in either direction (i.e., $c_{ij} = c_{ji}$). Typical performance related constraints include a limit on the maximum average queuing delay T_{max} in the network, and a limit on the number, or total length, of links in the path between origin and destination nodes. Protecting the network against link failures can be achieved by constraining the minimum degree of each node producing a k-connected network. For simplicity, we do not include constraints on the path length or connectivity of the network in our mathematical formulation of the problem. The ability to enforce these types of constraints is often included in the design of algorithms.

Viewing the MCMNDP as a multicommodity flow problem, each node in the network can be considered as the source of a commodity, varying volumes of which must be delivered to the other nodes in the network. Each commodity, $k \in K$, has a single source node s^k, a set of destination nodes $t_i^k \in (N - s^k)$ and a volume of traffic, r_i^k, to be delivered from s^k to each destination t_i^k. Let $r^k = \sum_{i \in X} r_i^k$ be the total traffic for commodity k. Also, let $\psi_u = \psi_{ij} + \psi_{ji}$ denote the total traffic flow on link u=(i, j), and φ_{ij}^k the total flow of commodity k on link (i, j). Using this notation the Minimum Concave Cost Multi Commodity Packet Switching Network Design problem is formulated as follows:

$$P = \left\{ \begin{array}{lll} \min \Phi(c) = \sum_{u \in U} \Phi_u(c_u) & & \\ \text{subject to} & & \\ \sum\limits_{\{j \ni (i,j) \in X\}} \varphi_{ij}^k - \sum\limits_{\{j \ni (j,i) \in X\}} \varphi_{ji}^k = \left\{ \begin{array}{ll} r^k & \text{if } i = s(k) \\ -r^k & \text{if } j = t(k) \\ 0 & \text{otherwise} \end{array} \right\} & & (1) \\ \lceil \psi_{\text{ij}} + \psi_{\text{ji}} \rceil \le c_{ij} & \forall (i,j) \in U & (2) \\ \text{Average Network Delay} \le \text{T}_{\text{max}} & & (3) \\ \psi_{ij} \ge 0,\ \psi_{ij} \ge 0 & \forall (i,j) \in U & (4) \end{array} \right\}$$

where $\lceil y \rceil_Q$ is a ceiling operator over the set of allowable link capacity values, Q. That is, $\lceil y \rceil_Q$ denotes the smallest value from Q, not less than y. Constraint (1) is standard node conservation of flow for each commodity k. Constraint (2) ensures that the capacity assigned to a link is greater than the traffic on the link. Constraint (3) employs Gerla and Kleinrock's expression [11] for the maximum average queuing delay, to ensure that the capacity assigned to each link is such that the average queuing delay does not exceed T_{max} (as specified by the network designer). Constraint (4) ensures non-negativity in the decision variables.

CHAPTER 3

LITERATURE SURVEY

In the literature, many variations of MCMNDP have been studied and solution procedures have been proposed both for static and dynamic cases. Circuit switching network design studies [19, 25, and 27] as well as packet switching network design examples [11,12,23] exist. In these works, the traffic for each origin destination pair is assumed to be known and deterministic. It is also assumed that the capacity utilization on every arc can be 100% for circuit switching links or at a desired level to meet the performance requirements for packet switching links.

Capacity expansion problems have also been studied within the context of MCMNDP [8]. An extensive literature survey can be found in [28]. The following paragraphs present a complementary literature survey for the MCMNDP.

Dutta and Lim [8] addressed the problem of deciding where, when and how much transmission capacity should be installed over a multi-period horizon, to meet traffic requirements at minimum total discounted cost, while maintaining an acceptable performance level. Their model allows traffic among existing nodes to increase, new nodes to be added to the network and its topology to change over time. This model is formulated as an integer-programming problem and a Lagrangian relaxation based solution method is proposed. Capacity and routing decisions are made jointly over time.

Amiri and Pirkul [1] studied a minimum cost, multicommodity network flow problem in which the total cost is piecewise linear, concave of the total flow along the arcs. They also extend the problem to the case of piecewise nonlinear, concave cost function. New mathematical programming formulations of the problems are presented. Efficient solution procedures based on Lagrangian relaxations of the problems are developed. This new formulation produces better results compared to [2]. In addition, experiments has shown that the piecewise nonlinear cost function estimates the discrete cost function more accurately than the piecewise linear functions.

Yaged [25] proposed a fixed-point algorithm based on successive approximation that converges in a finite number of steps to a local optimum. It is simple and computationally effective. However, the local minimum obtained is highly dependent on the starting solution, i.e., the algorithm strongly tends to produce an extreme point that, in sense, is "the closest" to the starting point. Therefore, the result is good if the starting solution is already a good approximation. This work has been extended to the dynamic routing problem in another study by Yaged [27].

Yaged's linearization method based on Kuhn-Tucker optimality conditions. Minoux [19, 20] proposed another optimality condition that is stronger than Kuhn-Tucker. He developed a greedy algorithm for the concave cost networks, which eliminates one uneconomical link at each iteration. An acceleration method for this greedy, called accelerated greedy method is also introduced.

In [11], Gerla and Kleinrock assumed link cost functions to be linear or concave to reflect the effects of economies of scale in the pricing of link capacity. By using a flow deviation technique linearization of concave link cost functions and continuous approximations to discrete capacity variables, Kleinrock and Gerla developed a Concave Branch Elimination (CBE) procedure to solve the Minimum Concave Cost Multicommodity Packet Switching Network design problem. A similar design procedure called the Cut-Saturation (CS) technique is also presented in [11]. Like the CBE procedure, it employs the flow deviation technique to route traffic in the network.

Another method proposed by Kleinrock and Gerla [11] is the Branch Exchange Method (BXC). This method starts from an arbitrary topological configuration and reaches a local minimum by means of local transformations (a local transformation, often called branch exchange, consists of elimination of one or more old links and the insertion of more new links). In [11], it has been shown than BXC method requires more computational time compared to CBE with a worse quality network solution.

Gersht and Weihmayer [12] formulate the TCFA problem following Kleinrock and Gerla [11] but develop a design procedure based on the greedy elimination of links. Gersht's algorithm assumes only that link cost functions are concave and continuous, without relying on the exact form of the function. Gersht claims that his procedure produces network designs whose cost is within a few percent of those produced by much more sophisticated algorithms in use within IBM.

While Gersht's algorithm performs well, its processing requirements are such that it is applicable for small network design problems only. Stacey, Eyers and Anido [23] present a Concave Link Elimination (CLE) procedure, based on Gersht's greedy link elimination procedure. This algorithm is shown to perform at least as well as Gersht's procedure and to be significantly faster than both the CBE and Gersht procedures. In addition, they formulate a lower bounding problem which they solve using a continuous branch-and-bound procedure to assess the quality of the design procedures.

Stacey, Eyers and Anido [23] claimed that the only methods applicable to concave topology capacity and flow assignment problem are Gerla and Kleinrock's CBE method and Gersht's greedy. Minoux accelerated greedy algorithm is evaluated as non-applicable approach because it is possible to have cycles in accelerated greedy method and the monotonocity assumption is not valid when the link cost function is concave. A comparison of CBE, CLE and Gersht greedy has been performed and according to results, CLE seems to be the most effective method.

A polynomial time dynamic-programming algorithm for MCMNDP has been proposed by Ward [29]. However, this algorithm is applicable to strong-series-parallel networks, which are encountered in inventory management and capacity planning, not in backbone telecommunication/computer networks. In this work a characterization of extreme flows in strong-series-parallel networks and an algorithm to find these extreme flows have been given. The computational time complexity of the proposed algorithm is $O(A(N+K))$, where A is the arc number, N is the number of nodes and K is the number of commodities in the network.

A fast design algorithm for mesh or distributed circuit switched networks, called MENTOR, is presented in [13]. The goal of the algorithm is not to find the optimal solution but a good one that can be used as a starting point for further optimization. Alternatively, the MENTOR algorithm could be embedded inside a design loop for designing hierarchical networks. Link costs are assumed to be concave functions of distance, stepwise concave with capacity, and incur a fixed installation cost. A variant of the MENTOR algorithm, called MENTour, is presented in [4]. The MENTour algorithm ensures that the network designs are 2-connected (i.e., minimum node degree of 2). This is useful when link speeds are much larger than traffic requirements, as the MENTOR algorithm tends to produce tree networks when this is the case.

Yan and Lou [30] have studied the MCMNDP for transportation, which can be extended for, have studied the MCMNDP for transportation, which can be extended for backbone telecommunication networks. They have employed simulated annealing and threshold accepting techniques to develop a number of heuristics. They have also designed a network generator to test heuristics. Although, the heuristics are very fast, they are highly depends on the initial solution.

Monma and Sheng [21] describe a packet network design and analysis (PANDA) model that captures the important features of different packet technologies. This model evolved from much iteration with technology developers and network planners over several years. The main contribution is a methodology for designing low-cost backbone packet networks with satisfactory performance that is both practical and useful. This methodology is useful for investigating cost/performance tradeoffs of various network capabilities and components, thus providing a means for identifying potential cost and performance bottlenecks for different packet technologies and to guide capability requirements for new technologies.

Chattopadhyay, Morgan and Raghuram [5] introduced a method to solve the minimum cost backbone design problem with multiple hop communication paths and time delay as performance criteria. This algorithm combines a branch and bound method, which is used to find the network topology, with the Ford-Fulkerson algorithm. After branch and bound, the Ford-Fulkerson algorithm is applied to determine the optimal paths between each pair of communicating nodes. A depth first search using a cost heuristic that is a linear function of the maximum and current depth in the tree is used. Algorithm has been compared with a branch and bound algorithm, which is known to find the optimum with a large computation time, and it has found that this method is much faster than branch and bound method, therefore applicable to large sized network.

For reliable network design, Cheng proposed a 1-FT (fault tolerant to 1 link-failure) backbone design method [6]. As networks become huge, the backbone layout design is essential to network performance and reliability. A 1-FT backbone can survive any 1-link failure. Cheng proposed an efficient method based on genetic algorithms to solve the problem. The backbone network is represented by a list of ordered links. He has shown that, the proposed algorithm find a sub-optimal solution for most cases.

Among the above methods, Yaged's Linearization, Minoux greedy and accelerated greedy methods are applicable to MCMNDP with circuit switching links. For the MCMNDP with packet switching links, Gerla and Kleinrock's concave link elimination, Gersht's greedy and Stacey's concave link elimination methods can be used. In the next chapter, details of these algorithms are given.

CHAPTER 4

EXISTING APPROACHES

In this chapter, previously proposed methods which are applicable to minimum concave cost multicommodity network problem and which are evaluated in this work are discussed in detail. In the first part, methods for circuit switching networks and in the second part, methods for packet switching networks are discussed. In these models, except Gerla and Kleinrock's method, discrete link cost functions are converted to continuous functions, optimal backbone is found and then continuous link capacities are converted to smallest upper discrete cost, as shown in Figure-4.1

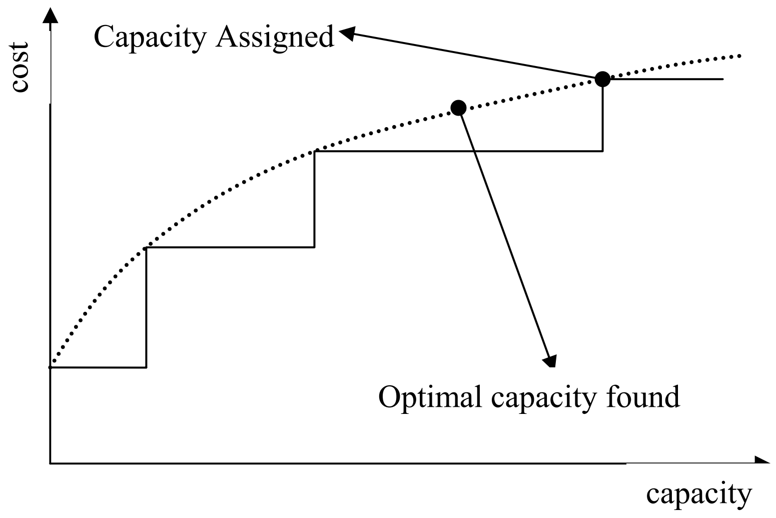

Figure 4.1 Discrete cost function usage

4.1 Circuit Switching Network Design Methods

4.1.1 Yaged's Linearization Method

Yaged [25] has studied the problem of selecting a path through the network for each point to point demand in order to minimize the total network cost. He has developed a special iterative technique based on Kuhn-Tucker necessary conditions for optimality to provide locally optimal solution by minimizing a concave function over a convex constraint set. A brief description of this technique is given in this section.

If each link cost is of the form $k_m \psi_u$, where k_m is a non-negative constant, the problem is reduced to finding the shortest path between each pair using k_m as the link length. However, the important point is to specify the *"length", k_m,* to be used for each link. Yaged has proposed a number of methods to assign lengths to each link.

Using the Kuhn-Tucker necessary conditions for optimality and the properties that the optimal routing should satisfy, the optimal path for routing found to be $\lambda_{ij} = \sum_{m \in p_{ij}^k} \Phi'_u(y_u)$ and the *ij* flow must be routed via a minimum length path. Therefore, it has been proposed to use the marginal cost $c_1 = \Phi'_1(\psi_1)$, as the link length for the shortest path calculation.

The algorithm consists of a two stage iteration in which the link cost estimates are assigned values in the first stage. In the second stage, a modified routing problem is solved to minimize the linearized network cost function . The links with a large total flow are given a smaller length and those with a small total flow are given a larger edge length for the next iteration. In the next iteration, the inexpensive links will draw more flow since some node pair paths will detour to a physically longer path in order to take advantage of decreased costs. Inversely, high cost links will be avoided. A link may become so costly that no flow traverses it and such a link will not be used for a route in later iterations. These uneconomical edges are eliminated from the network structure inherently. This iterative procedure is stopped when two successive iteration produce same flow patters, giving a fixed point.

For the link cost function displaying non-zero fixed cost c_f ,Yaged has found that use of *average cost pricing* instead of marginal pricing when $c_f >> 0$, where c_f is the fixed startup cost, will result in more economical network structures compared to marginal cost function. It is possible to use, $\partial\Phi / \partial\psi$, $\Phi(\psi) / \psi$, $\left(\Phi(\psi) + c_f\right)/\psi$ or $\left(\Phi(\psi) + 2c_f\right)/\psi$ as average cost. A useful property of the algorithm is that as the *steepness of the pricing curve increases, the number of links in the network usually decreases*. When solving a problem with fixed charges, it is better to use different pricing curves. By increasing the c_f of the average pricing curve, the number of links c_a be reduced. Once a network structure is obtained by average pricing, the algorithm should begin using marginal curve to provide small scale improvements in the cost.

When the link cost function has no fixed startup cost, it is still possible to use average cost as the pricing curve. Yaged has proposed the use of fictitious fixed cost $\left(\Phi(\psi) + K_f\right)/\psi$ as another alternative pricing curve. A detailed comparison of these pricing curves is given in Computational Study chapter.

4.1.2 Minoux Greedy Method

Minoux [19,20] has proposed a number of greedy methods for single commodity, multicommodity and nonsimultaneous flow network design problems. In this section a brief summary of the greedy method for the minimum cost multicommodity network design is given.

Minoux has observed that, although the method base on Kuhn-Tucker conditions proposed by Yaged [25] is simple and easy to implement and computationally efficient, the result obtained is highly dependent on the starting solution. As a consequence, the resulting local optimum will be good only if the starting solution was already a good approximation to the minimum cost solution, which is very difficult to solve. In order to overcome this fundamental difficulty, Minoux suggested a greedy method based on necessary optimality condition, which is stronger than the local optimality conditions provided by Kuhn-Tucker theorem.

Let $\psi^0 = (\psi_u^0)$ be any multicommodity flow on *G* and consider any edge *v=(i,j)* in *G* carrying strictly positive flow in this solution, i.e., such that $\psi_v^0 > 0$. Assign to the edges of *G* lengths $l_u > 0$ as follows:

$$\begin{cases} l_v = +\infty & \text{for edge v} \\ l_u = \Phi_u(\psi_u^0 + \psi_v^0) - \Phi_u(\psi_u^0) & \forall \text{u} \neq \text{v} \end{cases}$$

and denote by $L(\psi^0, v)$ the length of a minimum length chain joining *i* and *j* (the endpoints of edge *v*) in *G*. $L(\psi^0, v)$ can be interpreted as the minimum extra cost of globally rerouting all the flow that was running through edge *v* on a single alternative path in the network. Therefore, a necessary condition for ψ^0 to be a minimum cost solution is that:

$$\forall v \in U \text{ such that } \psi_v^0 > 0 : \Delta(v) = L(\psi^0, v) - \Phi_v(\psi_v^0) \geq 0 .$$

This necessary condition for optimality tells how to improve a solution that doesn't meet the necessary optimality condition. One of the best strategies which can be deduced from this result consists of a greedy type algorithm in which, at each stage k, where ψ^k is the current solution, all the differences $\Delta(v)$ are computed (for all v such that $\psi_v^k > 0$) and that edge which achieves the minimum Δ value is actually deleted, provided that $\Delta<0$, thus generating an improved solution ψ^{k+1}. The algorithm stops when the minimum Δ value is nonnegative, since this indicates that a solution meeting the necessary condition of theorem has been obtained. More formally stated, the greedy algorithm proposed by Minoux [20] is as follows

Step-1. Let ψ^0 be the starting solution ; $k \leftarrow 0$.

Step-2. At step k, let ψ^k be the current solution.

- For all $v \in U$ such that $\psi^k > 0$ compute $\Delta^k(v) = L(\psi^k, v) - \Phi_v(\psi_v^k)$
- Determine $\bar{v}$ such that, $\Delta^k(\bar{v}) = \underset{\substack{v\in U \\ \psi_v^k>0}}{Min}\{\Delta^k(v)\}$

Step-3. If $\Delta^k(\bar{v}) \geq 0$ STOP. (The current solution ψ^k meets the necessary optimality conditions.) Otherwise, let L^* be the minimum length chain linking the endpoints of $\bar{v}$ obtained at step-2. Let

$$\left\{\begin{array}{ll} \psi_u^{k+1} \leftarrow \psi_u^k & \text{if u} \notin L^*, u \neq \bar{v} \\ \psi_u^{k+1} \leftarrow \psi_u^k + \psi_{\bar{v}}^k & \text{if u} \in L^* \\ \psi_u^{k+1} \leftarrow 0 & \end{array}\right\}$$

Set $k \leftarrow k+1$ and return to step-2.

The worst-case complexity of the procedure can thus be easily evaluated. If *M* is the number of links in the initial solution, then at most *M* shortest path calculations are required at stage 1, *M-1* at stage 2, etc.. therefore, the overall number of shortest path computations is at most *M(M-1)/2*. Therefore, a worst case complexity of $O(M^2N^2)$ is achieved. In particular, in case the graph corresponding to the starting solution is complete or about to complete, this leads to $O(N^6)$ complexity. For large size networks, such a computational complexity requires too much time. Therefore, an acceleration technique to greedy algorithm is proposed by Minoux.

4.1.3 Minoux Accelerated Greedy Method

In order to decrease the computational complexity of greedy algorithm, Minoux proposed an acceleration to the greedy method. Minoux has observed that , as the greedy algorithm proceeds, once the Δ value associated with some link has become positive, then (apart from rather infrequent cases) the Δ value of this link doesn't take later on negative values again. This property indicates that, at each step , only the Δ values that were negative up to now, need to be recomputed. Exploiting this property, the number of shortest path computations decreases. Based on this observation, Minoux has made the following monotonocity assumption (M.A):

$$M.A. \quad \left\{ \begin{array}{l} \text{for each edge } v \in U : \Delta^{k+1}(v) \geq \Delta^{k}(v) \\ \text{by convention } \Delta = +\infty \text{ when the edge has been deleted} \end{array} \right\}$$

This monotonocity assumption holds in the case of minimum cost multicommodity flows when the cost function on the arcs are linear with fixed cost, but is not always satisfied in the case of arbitrary concave cost functions. The accelerated greedy algorithm can nevertheless be applied to this case since (even if there is no strict guarantee for that). Minoux has shown that, it most often produces the same final solution as the standard greedy algorithm. Moreover, in view of its tremendous efficiency, it can handle fairly large network synthesis problems. Algorithm steps for accelerated greedy algorithm are given below.

Step-1. Let ψ^0 be the starting solution ; $k \leftarrow 0$; $U^0 \leftarrow U$. For each link $v = (i, j) \in U$ such that $\psi_v^0 > 0$, determine the length $\overline{\gamma}(\psi^0, v)$ of the shortest i-j chain with respect to the lengths γ_u on the edges, on the partial graph of G induced by the edge subset $U^0 - v$. Set $\Delta(v) \leftarrow \psi_v^0 \overline{\gamma}(\psi^0, v) - (\delta_v + \psi_v^0 \gamma_v)$

Step-2. At the current step k, let ψ^k be the current solution.

Step-2.1. Determine v such that $\Delta(v) = \underset{u/\psi_u^k > 0}{Min} \{\Delta(u)\}$ and let *(i,j)* be the endpoints of v.

Step-2.2. Determine L^*, the shortest *i –j* chain with respect to the lengths γ_u on the edges on the partial graph induced by the edge subset:

$U_k - \{v\}$. If the length of L^* is $\gamma(\psi^k, v)$, let $\Delta' = \psi_v^k \overline{\gamma}(\psi^k, v) - (\delta_v + \psi_v^k \gamma_v)$. Set $\Delta(v) \leftarrow \Delta'$. If $\Delta' > \underset{\substack{u/\psi_u^k > 0 \\ u \neq v}}{Min} \{\Delta(u)\}$ return to Step-2.1.

Otherwise:

Step-2.3. If $\Delta' \geq 0$ STOP: the solution ψ^k cannot be further improved. (i.e., ψ^k is a greedy solution) Otherwise ($\Delta' < 0$) set:

$$\begin{Bmatrix} \psi_u^{k+1} \leftarrow \psi_u^k & \text{if } u \notin L^*, u \neq \overline{v} \\ \psi_u^{k+1} \leftarrow \psi_u^k + \psi_{\overline{v}}^k & \text{if } u \in L^* \\ \psi_u^{k+1} \leftarrow 0 & \end{Bmatrix}$$

Set $U^{k+1} \leftarrow U^k - \{v\}, k \leftarrow k+1$ and return to Step-2.

It has been reported that even for large problems, only 2 or 3 Δ values (instead of M) need to be recomputed at each stage with this strategy. The average speeding-up factor over the standard greedy algorithm which is about *M/3,* thus increases with the size of problem to be solved (resulting in an average computation time proportional to MN^2, instead of M^2N^2 for standard greedy algorithm.

4.2 Packet Switching Network Design Methods

4.2.1 Gerla and Kleinrock's Concave Branch Elimination

Gerla and Kleinrock [11,15] have proposed a number of packet switching network design methods. They have developed an average network delay model in terms of total network flow, link flow and the link capacity. In order to solve the design problem, they have divided the network design problem into three simpler sub problems. Based on this subproblems they have proposed a topology design method called Concave Branch Elimination(CBE). In this section, first a detailed summary of the delay model is given. Next, the three subproblems and finally the CBE method is summarized briefly.

4.2.1.1 Delay Analysis

For a packet switching network, average *"source to destination"* packet delay *T* is given as follows;

$$T = \sum_{\substack{j,k \\ j \neq k}} \frac{\gamma_{jk}}{\gamma} Z_{jk}$$

where Z_{jk} is the average packet delay from j to k, γ_{jk} is the average packet rate from source j to destination k and $\boldsymbol{\gamma}$ is the sum of all average packet rates in the network, $\gamma = \sum_{j,k} \gamma_{jk}$. Applying Little's result, the delay expression T becomes;

$$T = \sum_{u=1}^{m} \frac{\psi_u}{\gamma} T_u$$

where m is the number of links, ψ_u is the average packet rate (packet/s) on link u and T_u is the average queuing and transmission delay on the link u. However, it is not possible to measure T_u and ψ_u generally. In order to overcome this problem the following assumptions are made.

- External Poisson arrivals
- Exponential packet rate distribution
- Infinite nodal storage
- Static routing
- Independence between interarrival times and transmission times on each link.

Whit these assumption, average delay T_u on link u becomes, $T_u = \frac{1}{\mu c_u - \psi_u}$ where, $1/\mu$ is the average packet length (bits/packet) and c_u is capacity (bits/s) of link u. Using this average link delay, the average network delay becomes; $T = \frac{1}{\gamma}\sum_{u=1}^{m}\frac{\psi_u}{c_u - \psi_u}$. This delay model doesn't include the propagation delay on the channels and the nodal processing time.

Considering the delay expression, if link flow approaches the link capacity then delay goes to infinity. Therefore if both capacity and delay constraint are evaluated in the optimization, then *capacity constraints is implied by the delay constraints and can be disregarded.*

4.2.1.2 The Capacity Assignment (CA) Problem

The aim of CA is to find minimal cost link capacities satisfying the average network delay *T_{max}*, with given topology, requirement matrix *R* and link flow vector $\psi = (\psi_1, \psi_1, ..., \psi_n)$. For *linear link cost case* with positive startup cost, $\Phi_u(c_u) = d_u c_u + d_{u0}$ the optimal solution has been obtained by Lagrange multipliers. The optimal capacity and total cost is

$$c_u = \psi_u + \frac{\sum_{i=1}^{m} \sqrt{d_i \psi_i}}{\gamma T_{\max}} \sqrt{\frac{\psi_u}{d_u}} \qquad \Phi = \sum_{u=1}^{m} \left[d_u \psi_u + d_{u0} + \frac{\left(\sum_{i=1}^{m} \sqrt{d_i \psi_i} \right)^2}{\gamma T_{\max}} \right]$$

For the *concave link cost* case, the solution to CA problem can be found by linearizing the cost and solving a linearized problem at each iteration. This method reaches a local minimum. Kleinrock has shown that there exists a unique local minimum for concave cost function. Therefore, the solution found iteratively is the optimal solution.

4.2.1.3 The Routing (Flow Assignment) Problem

Given the network topology, link capacities $c = (c_1, c_2, ..., c_m)$ and requirement matrix R, the aim of Flow Assignment (FA) is to achieve a fixed routing policy minimizing the average network delay, $T = \frac{1}{\gamma} \sum_{u=1}^{m} \frac{\psi_u}{c_u - \psi_u}$

The above formulation is a convex multi commodity (MC) flow problem on a convex constraint set. Therefore, there is a unique local minima that is the global minimum. Previously an optimal method called *"Minimum Link Algorithm"* has been proposed by Chou and Frank [50]. However, this method requires too much computational time. A faster downhill search algorithm called *Flow Deviation (FD)* algorithm has been proposed. This method finds the optimal solution and is computationally efficient as the heuristic.

Flow Deviation

*Step-0***.** Let be a feasible starting solution. Let ***p***=0.

*Step-1***.**Compute **ø⁽⁰⁾** shortest route flow corresponding to $l_i^{(p)} = \left[\partial T / \partial \psi_i\right]_{\psi=\psi^{(p)}}, \forall i = 1,...,m$

*Step-2***.** Let $\overline{\lambda}_p$ be the minimization of delay expression $T\left[(1-\lambda)\phi^{(p)} + \lambda\psi^{(p)}\right], 0 \le \lambda \le 1$.. Let $\psi^{(p+1)} = (1-\overline{\lambda}_p)\phi^{(p)} + \overline{\lambda}_p\psi^{(p)}$.

Step-3. If the delay error is sufficiently small, $\left|T(\psi^{(p+1)}) - T(\psi^{(p)})\right| < \varepsilon$ then stop. $\psi^{(p)}$ is optimized within the given tolerance. If not, increment *p* and go to Step 1.

4.2.1.4 The Capacity and Flow Assignment (CFA) Problem

The CFA problem aims to find routing and capacity assignment of link to achieve minimal cost while satisfying average network delay. The CFA problem requires simultaneous optimization of routing and the link capacities.

When the link cost function is linear, i.e. $\Phi_u(c_u) = d_u c_u + d_{u0}$ it is possible to find the closed for expression for capacity c for a given flow ψ . Then the problem becomes minimization of cost with respect to design variable ψ as given below.

$$\Phi = \sum_{u=1}^{m}\left[d_u\psi_u + d_{u0} + \frac{\left(\sum_{j=1}^{m}\sqrt{d_j\psi_j}\right)^2}{\gamma T_{\max}} \right]$$

The Φ is concave over the convex polyhedron of feasible multicommodity flows. Therefore, there are several local minima corresponding to some corners of the polyhedron. FD algorithm with a bit modification can be used to find a local minima. In other words, FD performs a search on extremal flows, until it finds a local minima.

Modified Flow Deviation

Step-0. Let $\psi^{(0)}$ be a feasible starting solution. Let p=0.

Step-1. Let $\psi^{(p+1)}$ be the extremal flow corresponding to the following definition of equivalent length. $l_i^{(p)} = \left[\partial\Phi / \partial\psi_i\right]_{\psi_i=\psi_i^{(p)}} \forall i = 1,2,...,m,$

Step-2. If $\Phi\left(\psi^{(p+1)}\right) \geq \Phi\left(\psi^{(p)}\right)$ stop and $\psi^{(p+1)}$ is a local minimum. Otherwise *p=p+1* and go to 1. From the total cost expression, the equivalent length expression becomes;

$$l_i = d_i\left[1 + \frac{\sum_{j=1}^{m}\sqrt{d_j\psi_j}}{\gamma T_{\max}}\frac{1}{\sqrt{d_j\psi_j}}\right]$$

$\lim_{\psi_i \to 0} l_i = \infty$ .This implies that whenever the flow of link *i* is reduced zero at the end of an iteration, flow and capacity for that link will remain zero for all subsequent iterations. Therefore, uneconomical links tend to be eliminated by the algorithm. (observed by Yaged). The solution obtained by FD algorithm depends on the starting solution. To obtain better result perform FD algorithm several times for randomly selected flows.

However, it is not possible to express average network delay in terms of flow if cost function is concave. However, Gerla has showed that $\Phi(\psi)$ *is concave over* ψ and flow deviation algorithm can be applied for the concave cost case.

4.2.1.5 Topological Design

Concave Branch Elimination (CBE) Method

Start from a *fully connected* network and apply *modified Flow Deviation* algorithm until a local minimum is reached. Typically, FD algorithm eliminates uneconomical links and strongly reduces the topology. As two-connectivity is required the FD algorithm is terminated whenever the next link removal violated this constraint. The last two-connected solution is assumed to be the local minima. As the result of modified FD depends on the initial routing, to obtain several local minima, the FD algorithm is applied to several randomly chosen initial flows.

4.2.2 Gersht and Weihmayer's Joint Optimization Method

Instead of separating link capacity and facility selection from routing and topological design, Gersht and Weihmayer [12] have fully integrated the subproblems studied by Kleinrock and Gerla [11] to capture the very important coupling between them. They have proposed a delay formulation in terms of total network flow, number of nodes and link/node utilization, leading to a maximum number of link requirements in the network satisfying the average network delay. Using this maximum link number, a greedy method to design multicommodity network has been developed.

Let ψ^t_{mn} and ψ_m be the total flow on link *mn* and on node *m* respectively. Then the link and node utilizations should be bounded with values $p^t_L < 1$ and $p_N < 1$ respectively. Therefore the flow constrains can be formulated as;

$p_{mn}^t = \psi_{mn}^t / c_{mn}^t \leq p_L^t < 1$ and $p_m = \psi_m / B_m \leq p_N < 1$

$\psi_{mn}^t \leq p_L^t c_{mn}^t$ and $\psi_m \leq p_N \beta_m$

where c_{mn} is the capacity of link *mn* and B_m is the nodal capacity of node *m.*

For packet switching networks, the main performance criterion is the average network delay. The average network delay *D* is the sum of average link delay D_1 and average node delay D_2.

$$D = D_1 + D_2 \qquad D_1 = \frac{1}{\sum \lambda} \sum_{mn} \frac{p_{mn}^t}{1 - p_{mn}^t} \qquad D_2 = \frac{1}{\sum \lambda} \sum_{m} \frac{p_m}{1 - p_m}$$

where λ is the source to destination arrival rate. Taking the link/node utilization into account the delay expressions becomes;

$$D_1 \leq D_1^* = \frac{1}{\sum \lambda} \sum_{mn,t} \eta_{mn}^t \frac{p_L^t}{1 - p_L^t} \qquad D_2 \leq D_2^* = \frac{1}{\sum \lambda} N \frac{p_N}{1 - p_N} \qquad D < D^* = D_1^* + D_2^*$$

The main performance requirement is $D^* \leq D_0$ where D_0 is the maximum acceptable delay. This delay analysis linearizes the average network delay constraint and introduces *"maximum number of links"* constraint to the problem. The reason behind choosing equal node and link utilization for all the nodes and link is that, the optimal capacity assignment leads to a *balanced* network.

Gersht and Weihmayer [12] have proposed a greedy algorithm removing the most uneconomical link at each iteration. As the performance constraint *bounds the number of links* in the network, in some cases, although no further cost improvement is possible as the delay constraint is not meet new link reduction is performed resulting in cost increase. The algorithm stops whenever no further improvements are possible and performance constraint is meet. Below the details of the algorithm is given.

Define a topology T^v and its associated cost Z^v at each iteration *v* of the algorithm. A new topology T^v_{mn} with cost Z^v_{mn} is obtained by deleting candidate link *mn.* For each candidate link *mn* evaluate ΔZ^v_{mn} to find the cost difference. Denote the set of pre-elimination candidates by $\Delta Z^v = \{\Delta Z^v_{mn}\}$. At each iteration reduce the network cost with maximum $\Delta Z^v_{mn.}$ The algorithm steps is given below.

Step-0. Initialize T^0 with starting graph G, p_L and p_N incremental cost metric, Z^0 and iteration number v=1.

Step-1. Pre-elimination link *mn* generates T^v_{mv} by eliminating link *mn* in T^v

Step-2. Evaluate T^v_{mv} with respect to reliability (if more than one path for reliability is needed) and connectivity (two-two connectivity is still valid) constraints, if not meet goto Step-1. Else include T^v_{mv} in T^v .

Step-3. Find shortest path routing for T^v_{mv} *by* using the cost of route given in above definition. Compute the cost difference ΔZ^v_{mv} and add it to cost set ΔZ^v;

$\Delta Z^v_{mv} = Z^v - Z^v_{mn} \to \Delta Z^v$

In the shortest path problem the .measure is the cost of link startup and nodal costs. If every feasible link *mn* reduced goto Step4 else goto Step1.

Step-4. From the cost set ΔC^v find the link *mn* with max cost difference. The maximum cost difference can be a negative value. Select the link *mn* for deletion, If

- ✓ ΔZ^v_{mv} is positive means cost improvement.

- ✓ ΔZ_{mv}^{v} is negative (a cost increase) and there exists more links than the maximum acceptable link, due to delay constraint relaxation.

Selection of link *mn* in the first case is obvious because it results in cost improvement. In the second case, deletion of link *mn* results in delay improvement in spite of cost increase. Note that second case is selected if no further cost improvement is possible and delay constraint is not meet yet. One of the drawback of this algorithm is the computation time. As at each stage all the links are evaluated, requiring flow rerouting on the link, an only one link is deleted, this method is applicable to only small sized network.

4.2.3 Stacey, Eyers and Anido's Concave Link Elimination Method

Stacey, Eyers and Anido [23] has proposed an acceleration technique to al Gersht and Weihmayer's greedy. This new method, called Concave Link Elimination (CLE), is similar to Gersht's greedy, except at each iteration more than one link is deleted. Therefore this method is applicable to large sized network. Below a brief summary of the CLE is given.

The CLE procedure starts with all possible links in the network and eliminates subsets of links maximizing the reduction in network cost in each iteration while continuing to satisfy any connectivity or reliability constraints. The algorithm terminates when it cannot eliminate any more links without increasing the network cost or violating the capacity, performance, connectivity, or reliability constraints imposed on the design. The link cost structure in this method is proportional to link length, therefore CLE employs physical distance as the link length for shortest path routing. While Gersht uses a constant link utilization factor (supplied by the user) to assign link capacities, CLE procedure can assign link capacities using either CA procedure for concave link cost functions or Dutta's equivalent link utilization method.

To allow the algorithm to eliminate multiple *"unrelated"* links simultaneously, the algorithm has to evaluate *not only the effect of eliminating each link on the cost of the network, but also its effect on the traffic levels on other links in the network.* Using this information the algorithm can eliminate all links in each iteration of the search that reduce the cost of the network and do not affect the traffic load on each other.

Another improvement to Gersht procedure is that, if the elimination of a link will violate degree constraint, it should be in the final solution and this link will not be evaluated in the remaining of the algorithm. Below the CLE procedure is given.

Step-1. The initial network is assumed to be fully meshed. Calculate all O-D shortest pat, route traffic with this shortest path and assign link capacities using Dutta and Lims Equal Link Utilization [8] or CA [11]. Initial cost is Z and the set of required links is empty.

Step-2. (Candidate link elimination) Mark all links not in the set of required links as *"unevaluated"*. Select a link *e* from the set of unevaluated links and mark as *"evaluated"*. Remove link *e*, and r*ecalculate O–D paths as required (to save time, only those shortest paths affected need to be recalculated).* Evaluate reliability and performance constraints. If constraints are not satisfied, reinsert link *e*, *recalculate* O–D paths as required (if constraint violations indicate link *e* is required in final design add it to the set of required links) and go to step3.

Step-3. (Evaluation). Route traffic and assign link capacities. If the cost improvement observed, append the cost of the new solution, link *e* and the set of links affected by the removal of link *e* to a list of candidate link eliminations.

Step-4. (Link restoration). Reinsert link *e*, and recalculate O–D paths as required. If there are no more candidate link eliminations to be evaluated go to step 5, else go to step 2.

Step-5. (Link elimination). Start with two empty lists, one for the *set of links to be permanently eliminated,* call this *E*, and another for the *links affected by the elimination of the links in E*, and call this *AE*. Iterate through the list of candidate link eliminations generated in step 3 in ascending (having more improvement) order of resulting network cost . For each candidate link elimination, *if neither the link to be eliminated nor any of the links its elimination affects are included in either E or AE, add the candidate link to E and the set of links affected by its elimination to AE.* If, when the list of improving candidate link eliminations is exhausted, set *E* is empty (there are no links to eliminate) go to step 6. Otherwise, eliminate the links in *E*, *recalculate O–D pair shortest paths as required,* route traffic, assign link capacities and calculate the cost of the new solution *C*. Go to step 2.

Step-6. Assign link capacities using or Equal Link Utilization [8], calculate solution cost, *Z.*

CHAPTER 5

MODIFICATION ON MINOUX GREEDY ALGORITHMS

The greedy algorithms proposed by Minoux [19, 20] are based on removing a link and re-routing the flow over the deleted link via a shortest path between edges of the deleted links. In this way, the total network cost is reduced at each step and finally a local minimum is obtained. However, if implemented in its given form in [20], it is possible to have some extra and unnecessary capacity assignment, which results in an excess network cost. Therefore, it is possible to have an improved solution over the Minoux greedy solution.

In the Minoux greedy approaches, the aggregate flow on the deleted link is re-routed over a new shortest path between the nodes of the deleted link. However, this move brings the possibility to have a cycle after rerouting the aggregate flow over the deleted link and this problem is addressed earlier in [28] and later in [23]. Figure-5.1 illustrates how a cycle can be formed after re-routing. When link (2,3) is found to be deleted in the network by the Minoux algorithm, all the flow on link (2,3) is rerouted through the path (2,1), (1,5), (5,4), (4,3). The algorithm is stated in its original form to work on undirected graphs and therefore a flow originally going from node 1 to 4 through the path (1,2), (2,3), (3,4) is changed to use the path (1,2), (2,1), (1,5), (5,4), (4,3), (3,4). This path has unnecessary flows such as the one going first from 1 to 2 and then from 2 to 1 and the one going first from 4 to 3 and then from 3 to 4.

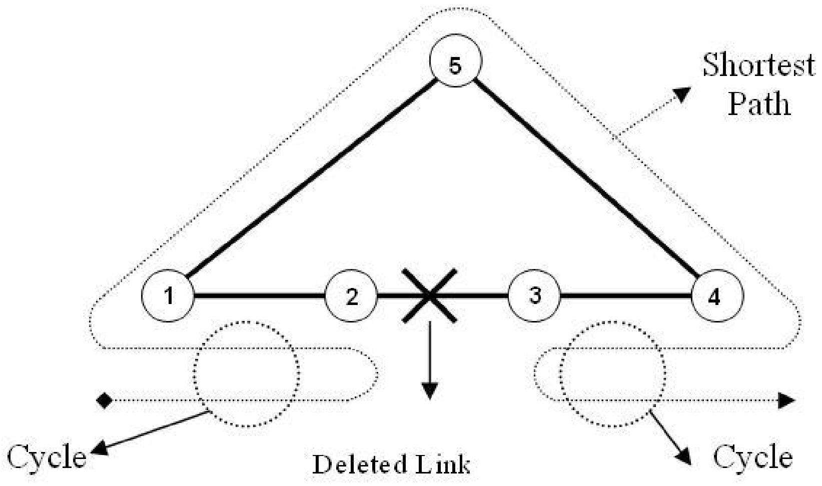

Figure 5.1: A cycle after rerouting

Minoux has actually realized the problem of excess network cost due to such cycles in his first paper [19] and he proposed a flow modification method based on algebraic manipulation as a remedy. However later, this modification scheme is shown to be not applicable in general and a more general modification to the original algorithm regarding the flow updating step has been proposed with an appropriate data structure.

The proposed modification scheme considers each single commodity flowing through the deleted link *u* separately. To find the best cycle free route, it first removes the corresponding flow from the current solution and then routes it via the shortest path between the source and destination of the commodity on a reduced graph G'. This reduced graph is formed from the links of the route with the cycle. Again, incremental cost is used as the link length, i.e. $l_u = \Phi_u(\psi_u + r^k) - \Phi_u(\psi_u)$. The algorithm steps proposed in [28] to replace Step-3 of the Minoux greedy method are given below. *CL(u)* denotes the set of commodities flowing through a given link *u* where the route L_k is the set of links on which the *k-th* commodity flows. The candidate link $\bar{v}$ is chosen for deletion in the current step of the Minoux greedy algorithm. In the following, $\oplus$ is an appending operation, i.e., adding and therefore augmenting to an existing list.

Step-3.1. Given the new partial route L^* at iteration *c*, do the following.

Step-3.2. Assign $\begin{Bmatrix} \psi_u^{k+1} \leftarrow \psi_u^k & \text{if u} \notin L^*, u \neq \bar{v} \\ \psi_u^{k+1} \leftarrow \psi_u^k + \psi_{\bar{v}}^k & \text{if u} \in L^* \\ \psi_u^{k+1} \leftarrow 0 & \text{otherwise} \end{Bmatrix}$

Step-3.3 Assign $\begin{Bmatrix} L_k \leftarrow \{L_k - \bar{v}\} \oplus L^* & \forall k \in CL(\bar{v}) \\ CL(u) \leftarrow CL(u) \oplus CL(\bar{v}) & \forall u \in L^* \end{Bmatrix}$

Step-3.4 $\forall k \in CL(\bar{v})$, i.e., for all commodities flowing through $\bar{v}$, detect the existence of a cycle. If a cycle does not exist then continue with the next commodity and if all the commodities in $CL(\bar{v})$ are proposed then assign $CL(\bar{v}) = \{\}$, increment *c* and go to step 2 of the Minoux algorithm to continue with the usual iteration. Otherwise (if a cycle exists), do the following;

Step-3.5. Assign $\begin{Bmatrix} \psi_u^{c+1} = \psi_u^{c+1} - r^k & \forall u \in L_k \\ CL(u) \leftarrow CL(u) - \{k\} & \forall u \in L_k \end{Bmatrix}$

Step-3.6.1. Form a reduced graph G' composed of edges in L_k

Step-3.6.2. Assign lengths $l_u = \Phi_u(\psi_u^{c+1} + r^k) - \Phi_u(\psi_u^{c+1}) \quad \forall u \in G'$.

Step-3.6.3. Find the shortest chain Γ_k^* between *s(k)* and *t(k)*

Step-3.7 Γ_k^* being the shortest chain found in step 6 between *s(k)* and *t(k),* assign $L_k \leftarrow \{\}$ and

$$\begin{Bmatrix} \psi_u^{c+1} = \psi_u^{c+1} + r^k & \forall u \in \Gamma_k^* \\ CL(u) \leftarrow CL(u) + \{k\} & \forall u \in \Gamma_k^* \\ L_k \leftarrow L_k + \{u\} & \forall u \in \Gamma_k^* \end{Bmatrix}$$

In the above method a shortest path calculation is required for each cycle detected. An effective method to search for cycles is also needed. Therefore it may be worth investigating the effect of not using shortest path computations but remove cycles as they are encountered directly. In the following paragraph, the pseudo code for the simple cycle search and elimination task is presented. The method just traverses the route of each flow on the deleted link and removes a cycle from the route if encountered. A doubly linked list structure is very suitable for the commodity route representation.

```
for (ALL commodities)
{ Point_Node= Commodity.Source;
  While NOT (Point_Node==Commodity.Destination)
  { Scan_Node=Point_Node.Next_Node;
    While NOT (Scan_Node ==Commodity.Destination)
    { IF (Scan_Node==Dummy_Node) //Cycle Detected
         {Dummy_Node=Point_Node;
         While NOT(Dummy-Node==Scan_Node)
      { Link[Dummy_Node][Scan_Node].RemoveCommodity(Current_Commodity)
        Dummy_Node=Dummy_Node.Next_Node; }
         Dummy_Node=Point_Node;
         Next_Node=Point_Node.Next;
        End_Node=Scan_Node.Next;
        While NOT(Next_Node==End_Node)
    {Dummy_Node=Next_Node;
     Next_Node=Next_Node.Next;
    Delete Dummy_Node;}
        Point_Node.Next=Next_Node;
       Scan_Node=Point_Node.Next;}
```

```
 ELSE
       Scan_Node= Scan_Node .Next;}
  }
Point_Node= Point_Node .Next;
}
}
```

The above method removes the cycle and routes the commodity with cycles over the original path of the corresponding commodity. The method proposed in [28] remove the cycle and routes the commodity with cycles over a reduced graph induced by the links on the commodity route with cycles. It is also possible to re-route such a commodity over the entire network topology. Such a re-routing requires a larger time consuming shortest path calculation and the cycles will be eliminated inherently. In order to compare the solution quality and timing requirements of re-routing over reduced graph and local cycle removing, a number of computational tests have been carried and results are presented in Chapter 8.2. The computational study revealed that there is no need to re-route the commodities over the reduced graph and removing cycles locally seems be sufficient.

CHAPTER 6

DISSAGREGATE LOCAL SEARCH METHOD

Minimum Concave Cost Multicommodity Network Design methods can be classified into two groups; those iteratively deleting a number of links at each step to reach a local minimum and those re-routing all the commodities at each step and deleting links inherently.

Minoux [19, 20] greedy methods, Gersht's [12] joint optimization and Stacey's [23] Concave Link Elimination methods are the members of the first group. They select one or more candidate links at each step for deletion and route all the commodities flowing through the candidate link/links. The iterations terminate when the deletion of a link is no more feasible because of not decreasing the network cost any more.

Yaged's [25] linearization method and Gerla and Kleinrock's [11] Concave Branch Elimination methods belong to the second group. These methods are based on re-routing all the commodities at each step using different link length criteria for the shortest path calculation. Uneconomical links are removed from the network inherently because if a link has zero flow, it will be never selected for a route of any commodity. If the routings of successive two iterations are exactly the same, the optimization process stops.

Rather than considering only the total aggregate flow ψ_u on each link, allowing simultaneous redirection of more than one commodity at each step, as in the algorithms mentioned above, it may be advantageous to use a disaggregate approach by considering a single commodity at a time. A similar approach for transshipment networks on directed graphs has already proved effective, where a partial separation of the flows on each link according to their origins was used [3, 9]. It is worth noting that flows in these works originating from the same source and distributed to various destinations were considered as single commodity (multi-terminal flow). In contrast, each flow corresponding to an origin-destination pair may also be regarded as a single commodity.

A greedy technique, called the Disaggregate Local Search, based on such an alternative single commodity definition has been proposed earlier [31]. This method is similar to the second group where only one commodity is handled at each step rather than the aggregate flow and hence being named as “Disaggregate”. In this approach, un-economical links will be eliminated inherently similar to the second group. The proposal and the results relating to the local optimality conditions of the approach are presented in the following paragraphs in its original form.

We start by few definitions. A chain $L = \{u_1, u_2, ..., u_q\}$ is a sequence of q links such that link u_r of sequence $(2 \le r \le q-1)$ has one common endpoint with link $u_{r-1}(u_{r-1} \ne u_r)$ and a second common endpoint with the link $u_{r+1}(u_{r+1} \ne u_r)$ [9]. The endpoint *i* of u_1 which is not adjacent to u_2, and the endpoint *j* of u_q which is not adjacent to u_{q-1} are called endpoints of the chain *L*. An *elementary chain* is such a chain that when it is traversed from one endpoint to the other, the same vertex is not encountered twice. A cycle is a subsequence of a chain whose endpoints coincide.

A flow pattern $\varepsilon = (\varphi^1, \varphi^2, ..., \varphi^{|K|})$ is called an *extremal flow pattern* if and only if for all source nodes *i*, the multi terminal flows originating from *i* form a tree with the common source *i* as root and the sinks as terminals [9].When not empty, the set of optimal solutions of problem contains extremal flow patterns. An implication is that, at an optimal solution each single commodity (disaggregate) flow φ^k of value r^k flows though an elementary chain between *s(k)* and *t(k)* (elementary chain property).

The disaggregate approach presented here considers only solutions satisfying the elementary chain property. It is worth noting that such solutions are not necessarily vertices of the constraint polyhedron (not a multi-terminal tree).

Let S_k be the set of all possible flow solutions φ^k corresponding to elementary chains only and let L_k denote the chain along which the k-th commodity flows. Then the redundant constraint $\varphi^k \in S_k$ can be added to problem *(P)* to obtain (P'). The two problems are equivalent in the sense that any solution of (P') is a solution for (P) and if (P) has a solution then (P') has one, too.

Given a solution $\varepsilon = \left(\varphi^1, \varphi^2, ..., \varphi^{|K|}\right)$ with $\varphi^k \in S_k \forall k$, a neighborhood $N(\varepsilon)$ can be defined as $N(\varepsilon) = \bigcup_{k \in K} N_k(\varepsilon)$ where

$$N_k(\varepsilon) = \left\{ \overline{\varepsilon} = \left(\overline{\varphi}^1, \overline{\varphi}^2, ..., \overline{\varphi}^{|K|} \right) : \overline{\varphi}^k \in S_k \text{ and } \overline{\varphi}^h = \varphi^h \text{ for } h \neq k \text{ and } h \in K \right\}$$

Definition 1: A flow pattern ε with $\varphi^{\mathrm{k}} \in S_k \ \forall k$, is a local optimum for problem (P) if $\phi(\varepsilon) \leq \phi(\overline{\varepsilon})$ for any $\overline{\varepsilon} \in N(\varepsilon)$, where ϕ is the total network cost.

To check for the optimality in Definition 1, we simply consider whether we can redirect any commodity k flowing through L_k via a new chain $\overline{L}_k$ such that the current objective function value is improved. Such a move generates a total cost difference of

$$\Delta_k(\overline{L}_k) = \sum_{u \in L_k \setminus \overline{L}_k} \left[\phi_u(\psi_u - r^k) - \phi_u(\psi_u) \right] + \sum_{u \in \overline{L}_k \setminus L_k} \left[\phi_u(\psi_u + r^k) - \phi_u(\psi_u) \right]$$

Let

$$\Delta_k\left(L_k^*\right) = Min_{\overline{L}_k \in S_k} \left\{ \Delta_k\left(\overline{L}_k\right) \right\}$$

$\Delta_k(L_k^*)$ is the decrease in total cost if $L_k^* \neq L_k$ and is zero if $L_k^* = L_k$.

Given $\varepsilon = (\varphi^1, \varphi^2, ..., \varphi^{|K|})$ with $\varphi^k \in S_k \ \forall k$ and a commodity *k,* L_k^* can be easily determined by solving the shortest path from s(k) to t(k) on $G = [X, U]$ with link weights $C_u \ \forall u \in U$, assigned as

$$C_U = \begin{Bmatrix} \phi_u(\psi_u) - \phi_u(\psi_u - r^k) & \text{if } u \in L_k \\ \phi_u(\psi_u + r^k) - \phi_u(\psi_u) & \text{if } u \notin L_k \end{Bmatrix} \quad (1)$$

Then $\Delta_k(L_k^*) = \sum_{u \in L_k \setminus L_k^*} C_u + \sum_{u \in L_k^* \setminus L_k} C_u$.

To see that the shortest path problem defined above does indeed lead to the desired result, argue as follows: withdraw the flow corresponding to the commodity *k* under consideration, then the cost of the flow on any link is given by equation 1 and the cheapest chain corresponds to the shortest path. If the solution thus obtained is cheaper than the current solution, then the saving is indicated by a negative $\Delta_k(L_k^*)$, but if the solution remains the same, then $\Delta_k(L_k^*) = 0$.

Letting $\Delta = Min_k \{\Delta_k(L_k^*)\}$, the following propositions then follow directly.

Proposition 1. $\Delta_k(L_k^*) = 0$ if and only if $\phi(\varepsilon) \leq \phi(\bar{\varepsilon})$ for any $\bar{\varepsilon} \in N_k(\varepsilon)$.

*Proposition 2***.** ε is an extremal flow pattern and is a local optimum by Definition 1 if and only if $\Delta = 0$.

Since the number of commodities $|K|$ may be as large as *n(n-1)/2,* a steepest descent is not computationally feasible. Instead, we employ the following non-greedy algorithm in which k' indicates the last re-routed commodity. It is worth noting that the resulting solution depends on the order of the commodities processed.

Algorithm steps for Disaggregate Local Search are given below.

Step-1. (Initialization) Given a network $G = [X, U]$ with pairs *s(t)-t(k)* and requirements $r^k \ \forall k \in K$, if initial solution ε^0 is not known then assign $\psi_u = 0, \forall u \in U$ and route each commodity *k* using the shortest chain L_k from *s(k)* to *t(k)* found after assigning weights $\phi_u(\psi_u + r^k) - \phi_u(\psi_u) \forall u \in U$.

Step-2. Set the *Improvement* =0. Assign $K'' = 0$; $k = 1$.

Step-3. (Evaluate and Reroute)Assign $C_u \quad \forall u \in U$ as in equation 1. Find the shortest chain L_k^* from *s(k)* to *t(k)* and compute $\Delta_k(L_k^*)$. If $\Delta_k(L_k^*) < 0$ then assign

$$\psi_u = \psi_u - r^k \quad \forall u \in L_k \setminus L_k^*, \psi_u = \psi_u + r^k \quad \forall u \in L_k^* \setminus L_k$$

and set $k' = k$. Replace old chain $\overline{L}_k$ with the new one L_k^*. Set *k=(k+1).* If $k > |K|$ then $k = 1$. Set the *Improvement* =0

Step-4. If *Improvement* variable is 1, goto step-2, else STOP.

As noted earlier, the solutions obtained at intermediate steps are not necessarily vertices of the constraint polyhedron. However, when the algorithm stops, the solution is an extremal flow solution, i.e., each multi-terminal flow origination from the corresponding source node forms a tree.

The solution quality of the local optimum reached highly depends on the evaluation order of the commodities in Step-3. There are 4 alternatives for the evaluation order.

- From largest requirement to smallest requirement, called descending order
- From smallest requirement to largest requirement, called ascending order
- In a random order
- In a steepest descent form (in improvement order)

The computational study in 8.4 has shown that, random order gives 2-3 % better quality network costs compared to descending order. Descending order gives 3-4% better solutions compared to ascending order. However, it is recommended to perform more than one random iteration to get better network cost solutions. As the number of iterations increases, the solution quality increases. The computational study has revealed that it may be sufficient to perform around 15 iterations for random order.

In the steepest descent form, called the *Disaggregate Local Search in Improvement Order*, the commodity that is re-routed at each iteration is the one having the maximum cost improvement after rerouting. Therefore this approach requires evaluation of all commodities and rerouting of the commodity with the largest improvement. Due to evaluation of all the commodities at each iteration, larger execution time is expected.

The computational study in Chapter 8 have shown that, the Disaggregate Local Search in Improvement order generates better solutions then other disaggregate approaches when the network size is small. As the network size increases, the solution quality decreases and it takes much more computation time.

CHAPTER 7

ADAPTATION of CIRCUIT SWITCHING METHODS TO PACKET SWITCHING MCMNDP

In chapter 2, problem formulation for both circuit switching network and packet switching network designs are given. The main difference between the circuit and packet switching network design problems is the average network delay requirement of the packet switching network due to queuing time of packets before transmission.

In this chapter, it will be shown that the methods proposed for circuit switching network design problems can easily be adapted to packet switching network design problems. Indeed, computational study has shown that, the adapted methods perform better the than previously proposed packet switching methods.

In this thesis two adaptation methods are proposed and evaluated. The first method uses the Capacity Assignment (CA) proposed by Gerla and Kleinrock [11] and details of this method is given in Section 4.2.1.2. The second one uses the Equal Link Utilization method proposed by Dutta and Lim [8]. Therefore, in the following section only the second approach is presented.

7.1 Equal Link Utilization

Dutta and Lim [8] proposed a multiperiod capacity model for backbone computer communication networks. The model is formulated as an integer programming problem and a Lagrangian relaxation based solution is used. The backbone computer communication networks studied is composed of packet switching type links.

The network design problem studied in [8] has an average network packet delay *D* as the performance. Each link can be regarded as an M/M/1 queue with the Gerla and Kleinrock's assumption of

- Poison arrival of external traffic at each node,
- exponential packet length distribution,
- infinite buffers and error free links.

Link service rate is c*(l)* and arrival rate is *f(l).* The resulting well known exponential for average packet delay gives rise to the following performance constraint.

$$\frac{1}{\sum_{i\in N}\sum_{j\in N} r(i,j)} * \sum_{l\in L}\frac{f(l)}{c(l)-f(l)} \le D \quad (1)$$

where *N* is the set of nodes, *L* is the set of links and *d(t)* is the maximum allowable average network delay.

Multiplying both sides of the delay constraints (1) by $\sum_i \sum_j r(i,j)$ results in:

$$\sum_{l\in L}\frac{f(l)}{c(l)-f(l)} \le D * \sum_{i\in N}\sum_{j\in N} r(i,j) \quad (2)$$

Dutta and Lim have made the assumption of equally distributing the value on the right-hand side over edges that exist in the network, resulting in a modified constraint. While it is possible, in principle, to have unequal distributions of the right-hand side over links, two factors argue strongly in favor of equal link utilization. Notice, first of all, that once the right hand side of (2) is distributed over *L*, equally and unequally, the ratio $f(l)/\left(c(l)-f(l)\right)$ is constrained for each $l \in L$. Equivalently, the utilization *f(l)/c(l)* is constrained. As noted before, delay on a link increases precipitously beyond a critical value if utilization become greater than this value.

To distribute the right-hand side of (2) unequally over, one needs to identify particular links that expect larger short term traffic fluctuations. These links can than be allocated a smaller fraction of the right hand side of (2) compared to links that are expected to experience small short-term traffic variations. The identification of such links, however, is very difficult. Therefore, it is appropriate to distribute the right hand-side of (2) equally over links.

$$\frac{f(l)}{c(l)-f(l)} \leq \frac{D * \sum_{i \in N} \sum_{j \in N} r(i,j)}{|L|} \quad \text{for } \forall l \in L \ \ (3) \ .$$

The right hand side of (3) is known from input values, and it will be represented by a single constant *ulink,* whose unit is packets. Since the link utilization, *f(l)/c(l)=ulink/(1+ulink)* , *ulink* can be viewed as a constraint on link utilization. Thus the modified performance constraint is

$$\frac{f(l)}{c(l)-f(l)} \leq ulink \ \text{ for all } l \in L, \qquad \text{where} \qquad ulink = \frac{D * \sum_{i \in N} \sum_{j \in N} r(i,j)}{|L|} \qquad \text{or}$$

equivalently, $\frac{ulink+1}{ulink} * f(l) \leq c(l)$ for $\forall l \in L$

The potential drawback of this algorithm is that the network may be overdesigned, i.e., it will have more capacity than it really needs to carry traffic demands while meeting the average network delay constraint. For instance, even a few link individually violate the simplified constraint by a small amount, the average network delay could still be less than the stipulated *D*. However, Dutta and Lim [8] has shown that in a large number of instances, this simplification caused relatively little excess capacity to be installed; the average network delay was not more than 10% below the desired limit.

In the following adaptation of the circuit switching network design method to the packet switching network with both Capacity Assignment[11] and Equal Link Utilization [8] methods are presented.

7.2 Adaptation Analysis

Networks that are generated using packet switching network design methods differ from that obtained using the circuit switching techniques by only with an excess capacity. Therefore, it is possible to meet the average packet delay requirement by adding enough excess capacity to each link of a network, which is already optimized by a circuit switching method. Capacity Assignment and Equal Link Utilization methods can easily be applied to such a network. Indeed, it is possible to embed these two methods to circuit switching network design methods to reach a minimized cost network with a desired average network delay.

In the linear link cost network, applying the adaptation method at the end of optimization and embedding it to the optimization method will give the same result. However, due to economies of scale, embedding the adaptation method to optimization process will give better result, especially when the link cost function is not same for all links. Therefore, in the remaining, adaptation is discussed for Modified Minoux and Disaggregate Local Search methods.

Both the modified Minoux greedy and Disaggregate Local Search use the incremental cost as the link length for the shortest path calculation. CA method can be applied after each routing to meet the average network delay requirement. While calculating the incremental cost use the link utilization assigned by CA to current solution. After re-routing perform CA again. Therefore the link weight in Disaggregate Local Search at iteration *c+1* becomes,

$$l_u = \phi_u\left((\psi_u^c + r^k) * \rho^c\right) - \phi_u\left(\psi_u^c * \rho^c\right) \forall u \in U \text{ where } \rho^c = \phi^c / \psi_u^c$$

Similarly for the Modified Minoux greedy, use the following link weight;

$$l_u = \Phi_u\left((\psi_u^c + \psi_\lambda^c) * \rho^c\right) - \Phi_u\left(\psi_u^c * \rho^c\right) \forall u \in U \text{ where } \rho^c = \phi^c / \psi_u^c$$

Adaptation of Modified Minoux greedy and Disaggregate Local Search methods using Equal Link Utilization is relatively simple. The only modification required is to change the link cost function to include the excess capacity. The cost function in Disaggregate Local Search method becomes,

$$\Phi_u\left(\psi_u\right) = \beta \psi_u^\alpha \left(\frac{ulink+1}{ulink}\right) + K_u \text{ where } ulink = \frac{D * \sum_{i\in N}\sum_{j\in N} r(i,j)}{|L|}$$

Likewise, at the *c+1* step of the Modified Minoux greedy method use the following cost function.

$$\Phi_u\left(\psi_u^c\right) = \beta \left(\psi_u^c\right)^\alpha \left(\frac{ulink+1}{ulink}\right) + K_u \text{ where } ulink = \frac{D * \sum_{i\in N}\sum_{j\in N} r(i,j)}{|L^c + 1|}$$

As mentioned before, CA is an iterative method requiring more computation time compared to Equal Link Utilization (ELU), which results in a very little computational time overhead for adaptation. Since ELU is much easier to apply and requires smaller computation time, ELU is chosen as the adaptation technique.

CHAPTER 8

COMPUTATIONAL STUDY

In this chapter, the computational study related to existing approaches, Modified Minoux greedy and Disaggregate Local Search are given. Comparisons of the algorithms are carried out for a number of network classes. In our study 25, 50 and 75 node networks are used where, a network size of 25 nodes is considered to be small. 50 and 75 nodes considered to be medium and large size networks, respectively.

The network traffic is considered as light when there is a possible network load of 10%. For moderate traffic and heavy traffic 25% and 50% load is assigned, respectively. In order to test the efficiency of the algorithms, 100% is used for full load traffic. Load is defined as the ratio of the number of commodities to all possible source to destination pairs.

The link cost function is a power law function $Cost(c_{ij}) = \beta c_{ij}^{\alpha} + K$, as shown in Figure 8.1. For small and medium sized networks 250 randomly generated network and traffic requirement is evaluated for each class. This number is 100 for large sized networks. The random network traffic , except the full load case, is distributed to nodes in *balanced* and *unbalanced* manner. When the same number of commodities are assigned to each node, then traffic is called *balanced*. If this number differs from node to node, the network traffic is *unbalanced*. In this work, the following method is used to generate random unbalanced network traffic.

i. Assign random commodities to randomly generated origin-destination pairs until all the nodes have exactly one commodity. If the number of nodes is odd, then one random node will have two commodities.

ii. Select random nodes as heavy traffic nodes. The number of these random nodes is 25% of the node number in the network.

iii. Generate random commodities with random origin-destination pairs from the set of heavy traffic nodes. Terminate this process when heavy traffic nodes are fully loaded or the total commodity number is the desired value.

iv. If the stopping condition is the first one, then assign random commodities with random origin-destination pairs from the set of light loaded nodes until the commodity number reaches to desired value.

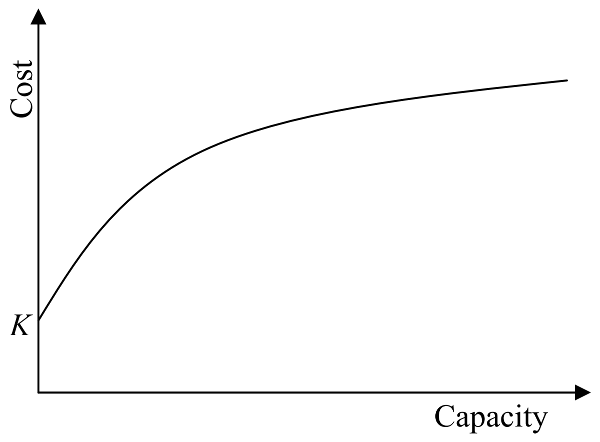

Figure 8.1: Link cost function used in the computational study.

In the rest of this chapter, the computational study results are given with 95 % confidence interval. The cost comparison tables in this chapter and Appendix represent average of percentage deviation of the corresponding method from the best result obtained for each randomly generated network. In the result tables, the best result associated with the network type is given in bold font. The network class is represented as *"N L P T"*, where

- N is the number of nodes in the network {25,50,75}
- L is the percentile of the possible traffic requirement {10%,25%,50%,100%}
- P is the concavity value α, {0.3, 0.7, Varying[0.3 to 0.6]}
- T is the traffic distribution to nodes, Balanced or Unbalanced

The computational study is carried on a Intel Pentium 4, 2.4GHz processor with 512 MByte RAM on a IBM PS/2 compatible PC running Microsoft Windows XP. The optimization programs are written in C++, and the timing information given in ms.

8.1 Yaged's Linearization Method Evaluation

63 network classes have been studied to see the effect of the selected pricing curve to reach a local minimum in Yaged's Method. These methods use the following pricing curves;

- Marginal cost , $\partial\Phi / \partial\psi$
- Average pricing , $\Phi(\psi)/\psi$
- Average pricing with fixed charge, $(\Phi(\psi)+K)/\psi$ where K is the fixed charge.

- Average pricing with fixed charge, $(\Phi(\psi)+2K)/\psi$
- Fictitious K_f with average pricing curve $((\Phi(\psi)+K_f)/\psi$

The computational study results in Table-A.1 have shown that, the quality of the pricing curves heavily depends on the cost function. If the network is composed of the same type of links, i.e. α is constant, then the average pricing curves yield the best network cost with any size of network. The cost of average pricing curve with $(\Phi(\psi)+2K)/\psi$, is 1-2 % better than other average pricing curves. The average pricing curve with fictitious startup cost gives network cost with a slightly different cost compared to $(\Phi(\psi)+K)/\psi$ average pricing curve. As the links of the networks studied have random startup cost, introducing fictitious startup cost doesn't change the result. Therefore, in Yaged's method the cost of startup is not important.

As α decreases, i.e. concavity increases, the difference between marginal and average pricing curves increases (up to 32 %). However as concavity decreases, i.e. link cost value becomes more linear, the difference decreases, but average pricing curves show better results.

This computational study shows that, as the steepness of the pricing curve increases, the number of links in the network decreases, resulting in lower network cost. In other words, the cost of average pricing functions can be ordered as follows;

$$(\Phi(\psi)+2K)/\psi \quad < \quad (\Phi(\psi)+K)/\psi \; < \; \Phi(\psi)/\psi$$

This property has also been mentioned by Yaged [25].

The result of fictitious fixed charge is worse than $(\Phi(\psi)+2K)/\psi$, in all cases when α is constant. As a result, $(\Phi(\psi)+2K)/\psi$ is the best pricing curve for all α and network size as long as α is constant.

However, when the network is a hybrid link network; marginal pricing curve gives the best solution. The average pricing curve gives around 70% worse result, which is very large.

As a result, when using Yaged linearization it can be recommended to use average pricing curve $\left(\Phi(\psi)+2K\right)/\psi$ when α is constant and marginal pricing curve $\partial\Phi/\partial\psi$, when α is varying.

8.2 Modified Minoux Greedy Method Evaluation

As mentioned in Chapter 5, there are two alternatives to get rid of the cycles in Minoux greed algorithms. In the first approach, the algorithm search for a cycle on each flow route. If a cycle is found, the route of the flow is modified by just removing cycle without any rerouting. Other path of this route remains unchanged.

In the second approach, the algorithm search for a cycle on each flow route. If a cycle found, the traffic for that flow is rerouted over the sub-network, which is composed of the node and link on the flow route, using the shortest path. During calculation of the shortest path, incremental link cost is used as the cost measure.

In order to find the difference of these two approaches, 42 network classes are evaluated using these two methods. The computational results in Table-A.2 have shown that, there is a very little solution quality between these two methods. The difference never exceeds 1 %. Considering the computational time, as the network size and flow size increases, more computational time is required for the method using rerouting over the sub-network, because rerouting process requires a shortest path calculations for each flow with cycle. As a result it is better to use the method, which just removes cycle from the path without rerouting the flow over the sub-network.

8.3 Disaggregate Local Search Method Evaluation

As mentioned before, there are four possible approaches in Disaggregate Local Search Algorithm. These approaches are;

- Disaggregate Local Search in largest commodity first order
- Disaggregate Local Search in smallest commodity first order
- Disaggregate Local Search in random commodity order
- Disaggregate Local Search in maximum improvement, i.e. steepest descent, order.

Disaggregate Local Search in largest commodity first order, as its name implies handles the commodities in a sorted order from largest demand to smallest one. Similarly, Disaggregate Local Search in smallest commodity first order handles commodities starting from the smallest demand to the largest one. Disaggregate Local Search in random commodity order, handles the commodities in random order at each iteration. Disaggregate Local Search in maximum improvement commodity order, calculates the improvement due to commodity rerouting at each iteration, then selects the commodity having the maximum improvement with rerouting.

Computational study has shown that it is better to perform more than one algorithm iteration for Disaggregate Local Search in random order. In order to find how much iterations are enough, a computational study on Disaggregate Local Search in random order is performed for 5,10,15,20 and 25 iterations. The results have shown that it is better to perform around 15 random iterations to get better networks when both cost and computational time is considered. Comparison of the results for various numbers of iterations is shown in Table-8.1.

Table-8.1: Solution quality comparison for some number of iteration in Disaggregate Local Search in random order.

5 iterations	10 iterations	15 iterations	20 iterations	25 iterations
2.21	1.44	0,94	0,85	0,84

63 network classes have been evaluated in order to compare the approaches in the Disaggregate Local Search methods. Computation results in Table-A.3 have shown that the Disaggregate Local search in largest order, smallest order and random order has the same order of quality. In most cases the quality difference between these three approaches is around 1%. Generally, these algorithms can be ordered according to their qualities as follows;

Disaggregate Random > Disaggregate Largest > Disaggregate Smallest

The fourth method, Disaggregate Local Search in Maximum order gives almost the same result with Disaggregate Local Search in Largest order, except in Small network classes. It performs better than Disaggregate Local Search in Largest order.

The final issue for the Disaggregate Local Search algorithm is to find the starting point. When compared to other heuristics (like Minoux greedy), Disaggregate Local Search algorithm doesn't eliminate links at each iteration. The uneconomical links having zero flow are removed at the end of algorithm. Therefore the starting point determines the quality of the local minimum. Two possible starting points are considered in the computational study. In the first case, the flows are routed using shortest path based on physical distance of each link. In the second case, routing is performed based on hop count. Computational results have shown that it is better to use physical distance as the starting point, except the case where the network is fully loaded and the link concavity value is varying. In this case, it is better to use hop count as the starting solution [Table A.3].

As a result Disaggregate Local Search in Random order around 15 iterations gives the best result, in general.

8.4 Circuit Switching Methods Comparison

In order to compare the performance and computational time requirements of the circuit switching methods for MCMNDP, 63 network classes have been examined. Computational study has shown that, Modified Minoux greedy algorithm gives better quality solutions in all network classes with very little computational time increase compared to original Minoux greedy methods. When the links are highly concave (α=0.3), Modified Minoux greedy algorithm shows 3-4% performance improvement over the greedy algorithms. When the link cost becomes more linear (α = 0.7) the differences decreases to around 1%. In the network classes with varying link cost concavity (α=varying) Modified Minoux greedy algorithm generates network with 4-5% better network cost compared to Minoux greedy algorithm.

The acceleration method to Minoux greedy algorithm, called Minoux accelerated greedy algorithm has also been evaluated. The modification to Minoux greedy algorithm is also applied to the Minoux accelerated greedy algorithm, called Modified Minoux accelerated greedy algorithm. We expect to have a faster algorithm with a cost of solution quality degradation w.r.t. Minoux greedy algorithm. Computational study has shown that, when the link cost concavity is high (α = 0.3) and link cost concavity is varying (α = varying) the solution quality of both Minoux greedy and Minoux accelerated greedy algorithm are almost the same. As a result of this, there is a little difference between the computational times. Minoux accelerated greedy algorithm works a bit faster. However, when the link cost function becomes more linear (α=0.7), Minoux accelerated greedy algorithm becomes faster (requires almost half of the computational time) compared to Minoux greedy algorithm. In such networks, Minoux greedy algorithm has 1-2% better network costs.

The computational results regarding the comparison of Minoux greedy and Minoux accelerated greedy algorithm are also valid for Modified Minoux greedy and Modified Minoux accelerated greedy algorithms. The performances of the circuit switching methods are discussed in the following sections for different network classes. Cost comparison of methods is shown in Table-A.4. Table-A.5 shows the computational time required. Table 8.2 presents the recommended network design technique deduced from the computational study performed within the context of circuit switching network design. The entries in the table have been derived from the computational study results presented in Table-A.4 and Table-A.5.

8.4.1 Networks with Large Economies of Scale ($\alpha = 0.3$)

Computational results have shown that, when the network is composed of links with highly concave link cost functions, the Modified Minoux greedy algorithm gives the best quality networks in all size of networks and traffic conditions. The Accelerated Modified Minoux greedy algorithm gives the same result as the Modified Minoux greedy algorithm with a little bit smaller computational time.

Disaggregate Local Search algorithm gives network solution with around 20% larger network cost. This solution quality result is valid for almost all network sizes and traffic, except the small network with light traffic case. For that case, the solution quality is around 6% larger than the best solution.

For the Yaged’s Linearization, as the network traffic increases, better quality networks are obtained. However, the networks obtained with Yaged’s Linearization has 8-10% larger network cost compared to best solution in fully loaded networks. This quality decreases up to 20% in lightly loaded networks.

Table 8.2 Recommended circuit switching method for each network class

Network Size	Traffic Load	Economies of scale (α)	Method
Small	Light	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Disaggregate Local Search
	Medium	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Disaggregate Local Search
	Large	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Disaggregate Local Search
	Full	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Modified Minoux Greedy
Middle	Light	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Disaggregate Local Search
	Medium	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Modified Minoux Greedy
	Large	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Modified Minoux Greedy
	Full	Low	Modified Minoux Greedy
		High	Yaged Linearization
		Varying	Disaggregate Local Search
Large	Light	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Modified Minoux Greedy
	Medium	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Modified Minoux Greedy
	Large	Low	Modified Minoux Greedy
		High	Disaggregate Local Search
		Varying	Modified Minoux Greedy
	Full	Low	Modified Minoux Greedy
		High	Yaged Linearization
		Varying	Modified Minoux Greedy

When the computational time of the algorithms are compared, Yaged's Linearization is the fastest algorithm. The Modified Minoux greedy algorithm requires around 40-50 times more computational time compared to Yaged algorithm. Disaggregate Local Search Algorithm, which is the slowest algorithm, requires around 3 times more computational time compared to Modified Minoux greedy algorithm.

8.4.2 Networks with Small Economies of Scale (α = 0.7)

When the network is composed of links having nearly linear cost function near to linear (α = 0.7), computational results have figured out that, Disaggregate Local Search Algorithm gives best the result, in almost all network sizes and traffic conditions.

The solution quality of the Modified Minoux greedy algorithm in this network class decreases as the network size increases. In small networks its solution quality is 5% larger than the best solution. When the network is middle sized, the quality decreases to 10% and as the network size becomes large, solutions are about 15% larger than the best solution.

Yaged's Linearization method performs well in this network class, as the network becomes heavily loaded. For small sized networks, Yaged and the Disaggregate Local Search generate almost the same quality solution. As the network size becomes middle and large, Yaged performs a bit better than the Disaggregate Local Search. When the network is moderately loaded, the solution of Yaged is around 2-3% larger than the best solution and this value increases up to 8-10% when the network is lightly loaded.

Again, Yaged's Linearization is the fastest algorithm. Therefore, in this network class, it is recommended to use Yaged method when the network is heavily loaded.

The Modified Minoux greedy algorithm requires around 20-30 times more computational time. Disaggregate Local Search algorithm requires 4-6 times more computational time compared to Modified Minoux greedy algorithm [Table A.5].

8.4.3 Hybrid Networks (α =varying)

In this class of networks, according to computational results, it is not possible to indicate a method as the best for all network sizes and traffic. In some cases, Disaggregate Local Search algorithm is better, whereas Modified Minoux greedy algorithm gives best result in most of the cases.

For small and middle sized networks, Modified Minoux greedy algorithm gives better results as the network traffic increases. Network cost with Modified Minoux greedy algorithm is around 80% larger than the best result in small networks with light traffic. However, as traffic increases the quality increases to 10% in moderate networks and 1% in fully loaded network, in which the Modified Minoux greedy algorithm is best. For large size networks, Modified Minoux greedy algorithm gives the best result all the time.

Disaggregate Local Search algorithm generates high cost networks as the network becomes heavily loaded. When the network is lightly loaded, this algorithm gives the best result in small and medium size networks. As the network traffic increases, the cost of network increases around 7% compared to the best result. However, as the network becomes fully loaded, the cost of the network decreases down to 3-4% compared to best solution and even gives the best result for middle sized networks.

The computational results has shown that, as the network traffic increases Yaged's Linearization method gives better quality network. However, the cost of networks obtained by Yaged's Linearization method is around 10% larger than the best result. This value increases to 20% when the network traffic becomes light.

The computational time requirement of the methods, are similar to other network classes. Yaged's Linearization is the fastest. Modified Minoux greedy requires 40-50 times more computational time and the Disaggregate Local search method requires 2-3 times more computation time compared to Modified Minoux greedy.

8.5 Packet Switching Methods Comparison

In order to evaluate the performance of packet switching network design algorithms 63 network classes have been evaluated. The packet switching methods evaluated here are;

- Concave Branch Elimination (CBE) proposed by Gerla and Kleinrock [11]
- Gersht's [12] Joint Optimization approach
- Concave Link Elimination (CLE) proposed by Stacey [23]
- CLE Incremental, a modification to CLE
- Adapted Modified Minoux Greedy Algorithm
- Adapted Disaggregate Local Search Algorithm.

The Disaggregate Local Search Algorithm is repeated 15 times for random commodity selection approach.

CBE method is repeated 30 times and the counter re-started from one when a better cost network is achieved.

In CLE [23] while rerouting the flows, the shortest path is calculated using the physical length. However, in order to make a comparison of physical length and incremental length used by Minoux[19,20], CLE Incremental method is evaluated by changing the length to incremental cost in shortest path calculation.

The network classes studied can be grouped according to the concavity of link cost function. Computational results are shown in Table-A.6 and computational time requirements are given in Table-A.7 The recommended packet switching network design method for each network class is tabulated in Table-8.3 and these entities are derived from Table-A.6 and Table-A.7.

8.5.1 Networks with Large Economies of Scale (α = 0.3)

When the network is composed of links having the same concave link cost function resulting strong economies of scale, computational results has shown that Gersht's greedy gives the best result for all network size and traffic conditions.

The Adapted Modified Minoux greedy algorithm, which is around 7-8 times faster than Gersht's greedy gives very nice quality networks. When the network is lightly loaded, the cost of networks obtained with this algorithm is around 0.5-1% larger than the best quality. As the network traffic increases, the quality of this approach is a bit degraded. The cost of the networks is around 1-1.5% larger than the best result when the network is fully loaded.

The cost of networks generated by CLE decreases as the network traffic increases. When the network is lightly loaded, cost is around 5-7% larger than best cost network. As the network traffic increases, this value decreases down to 1-2%. CLE algorithm is around 3 times faster than Gersht's greedy when the computational time is considered.

Table 8.3 Recommended packet switching method for each network class

Network Size	Traffic Load	Economies of scale (α)	Method
Small	Light	Low	Gersht greedy
		High	Adapted Disaggregate Local Search
		Varying	Adapted Disaggregate Local Search
	Medium	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Disaggregate Local Search
	Large	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Disaggregate Local Search
	Full	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Disaggregate Local Search
Middle	Light	Low	Gersht greedy
		High	Adapted Disaggregate Local Search
		Varying	Adapted Disaggregate Local Search
	Medium	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Modified Minoux Greedy
	Large	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Modified Minoux Greedy
	Full	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Disaggregate Local Search
Large	Light	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Modified Minoux Greedy
	Medium	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Modified Minoux Greedy
	Large	Low	Gersht greedy
		High	CLE incremental
		Varying	Adapted Modified Minoux Greedy
	Full	Low	CLE
		High	Adapted Disaggregate Local Search
		Varying	Adapted Modified Minoux Greedy

The modified version of CLE algorithm, CLE Incremental method generates around 1% larger cost networks compared to CLE method in all network cases. Indeed, the computational time required for this method is around 1.5 – 2 times larger than CLE.

Concave Branch Elimination, which is the fastest method in this class of networks generates networks with 40-50% larger cost compared to the best network in lightly loaded network. As the network traffic increases, CBE generates better cost networks, however, the cost of networks is still 20-30% larger than the best quality.

The adaptation of Disaggregate Local Search Algorithm, generates larger cost networks in this class of networks as the network size increases. For small sized networks, cost is around 15-20 % larger than best. This value increases to 20-25 % for middle size networks. For large size network, 25-30 % larger cost networks are generated. The computational time required for Adapted Disaggregate Local Search Algorithm is about the same with that of CLE algorithm.

As a result, it may be recommended to use Gersht' greedy if computational time is not problem. For a faster approach, Adapted Modified Minoux greedy algorithm is the best choice.

8.5.2 Networks with Small Economies of Scale ($\alpha = 0.7$)

For this class of networks the solution quality of Modified Concave Link Elimination, namely CLE Incremental gives the best result in all network classes, except small size networks with light traffic load. In these networks, Adapted Disaggregate Local Search algorithm gives a bit better result.

CLE method generates networks with very little larger cost compared to the best result, which is around 0.1-0.5 %. The computational time required is a bit shorter compared to CLE Incremental.

Another good algorithm for this class of network is the adaptation of Disaggregate Local Search Algorithm. There is a cost increase in the order of 1–2 % when Adapted Disaggregate Local Search algorithm is used. As mentioned before, it gives 1.5% better quality networks at small network sizes with light network traffic. The algorithm requires almost half of the time compared to CLE incremental at fully loaded networks. As the network traffic decreases more computational time is required.

CLE method is better than Gersht's greedy in this class. Gersht's greedy generates around 3 % larger cost network within around four times more computation time.

The quality of CBE algorithm increases as the network becomes heavily loaded. For lightly loaded networks the cost of CBE is around 50% larger than the best result. As traffic demand increases, this value decreases down to 20% in fully loaded networks. Although this algorithm is the fastest method, it is not recommended for this class of network due to its quality.

Another bad quality algorithm in this class of networks is the adaptation of Modified Minoux greedy algorithm. The algorithm quality decreases as the network size increases. For small networks, around 6% larger cost networks are generated. This value increases up to 16% in larger networks. Computational time required for this method is around quarter of time required for CLE Incremental.

As a conclusion, it is recommended to use CLE or CLE Incremental in this class of networks.

8.5.3 Hybrid Networks (α = varying)

The adaptation of Modified Minoux greedy and Disaggregate Local Search Algorithm are the best quality approaches in this class of networks. The size and traffic density of the network determine which method to use among these two algorithms.

Adapted Modified Minoux greedy algorithm gives better results as the network traffic increases. For small and middle sized networks, cost of the networks generated with Adapted Modified Minoux greedy is larger than the Adapted Disaggregate Local Search. Network cost is about 80% larger than best in small networks with light traffic. However as the network traffic increases, the cost difference with the best decreases down to 10% in moderate networks and %1 in fully loaded networks, in which case Modified Minoux greedy algorithm is best. For large size networks, Modified Minoux greedy algorithm gives the best result all the time.

Adapted Disaggregate Local Search generates high cost networks as the network becomes heavily loaded. When the network is lightly loaded, this algorithm gives the best result in small and medium sized networks. As the network traffic increases, the cost of network increases up to around 7% compared to best. However, as the network becomes fully loaded, the cost difference with best decreases down to 3-4% and even gives the best result for middle sized networks.

CLE method doesn't perform well in this class of networks according to computational study. It generates around 35% larger cost networks when the network size is small. As network size increases, quality becomes a bit better. However, the cost is still 20% large in large sized networks. Indeed as the network size increase, the computation time required increases compared to adaptation of Modified Minoux greedy and Disaggregate Local Search algorithms.

The modification to CLE, CLE Incremental has significantly better quality in this class of networks. CLE Incremental has around 15% better quality networks compared to CLE algorithm, with around two times computational time requirement compared to CLE. However, CLE Incremental also has 15% larger cost compared to best result in small sized networks. Like CLE, its quality becomes better as the network size increases. However, the cost is still 8% large in large sized networks.

Similarly, Gersht's greedy is not suitable in this class of networks because this method is very similar to CLE except it removes only one link at each iteration. Therefore, the computational time required is larger than CLE and CLE Incremental. The solution quality of Gersht's greedy is in between CLE and CLE Incremental.

With the increasing network size, the network cost quality increases with CBE method according to computational results. Although this method is a very fast algorithm, the quality is around 30% larger in small size network and 15% in middle sized networks compared to best results. This value decreases down to 10% for large sized networks.

If the network is composed of different concavity link cost functions, it may be advised to use adaptation of Modified Minoux greedy and adaptation of Disaggregate Local Search algorithms. The network type determines which algorithm to use according to the computational study.

CHAPTER 9

CONCLUSIONS

Concave cost multicommodity network optimization is quite important for many application areas such as in backbone telecommunication networks and in transportation planning. and is expected to remain so as long as strong economies of scale continues to exist. A number of methods to find the true optimal solution has been proposed but these methods are applicable only to small sized problems due to large computational time requirements. In the literature a number of greedy methods and optimization techniques based on Lagrangian Relaxation have also been proposed. The latter methods still require large computational time and therefore, heuristic techniques are usually employed for most concave cost multicommodity flow problems.

In this thesis work, methods for the MCMNDP with circuit and packet switching links are investigated and empirically evaluated. For the former network type with circuit switching links only, Yaged's linearization scheme and Minoux's greedy and accelerated greedy methods are considered. Previously, Minoux greedy and accelerated greedy methods have been demonstrated to create excess flows if used in their simplest forms. It has also been demonstrated that the original proposal for the removal of the excess flow leads to anomalies in the general concave cost case. In this study, the modification proposed for Minoux greedy methods has been evaluated in detail and proved to effective.

For the second network type with packet switching links only, Gerla and Kleinrock's Concave Branch Elimination method, Gersht's greedy and Stacey's Concave Link Elimination methods have been investigated Using Dutta and Lim's equal link utilization method, it has been shown that the methods for circuit switching network design can easily be extended to the packet switching case. This is the first time that the two groups of algorithms are compared with each other. In some problem classes, the proposed adaptation proved to be quite effective.

A Disaggregate Local Search, which considers a single commodity at each iteration, has also been evaluated for both circuit and packet switching networks. An extensive computational study has been carried out to evaluate the performance of the above techniques. The results reveal that compared to other algorithms, both the modified Minoux algorithm and the Disaggregate Local Search proved to be the most successful in finding good solutions in most cases.

The present work has also invalidated an earlier claim in [23] "that the Minoux greedy is not applicable to the concave cost network design". Our study has shown that, with the proposed modification Minoux greedy method has even better performance over CLE, which was proposed in [23].

One important result of our computational study is that the best method for the network design depends on the size, traffic and cost function of the network. The best technique for each class of problems employed in this work has been pointed out and tabulated in Chapter 8.

Further works remains for the optimization of real networks with the methods discussed in this work. Different combinations of the above heuristics in an integrated approach may be investigated. An integrated tool may also be developed which incorporates all the techniques and applies the appropriate ones with a higher decision level to get the best solution.

REFERENCES

[1] Amiri, A., and Pirkul, H., "New formulation and relaxation to solve a concave-cost network flow problem", *The Journal of Operational Research Society*, Vol 48, No:3 (1997) 278-287.

[2] Balakrishnan, A. and Graves, C.S., "A composite algorithm for a concave-cost network flow problem", *Networks* 9 (1978) 175-202.

[3] Bazlamaçcı, C.F., and Hindi, K.S., "Enhanced adjacent extreme-point search and tabu search for the minimum concave-cost uncapacitated transshipment problem", *Journal of Operations Research Society* 47 (1996) 1150-1165.

[4] Cahn, R.S, "MENTour: An algorithm for designing reliable high-speed data networks", *Canadian Journal of Electrical and Computer Engineering* 20(3) (1995) 101–103.

[5] Chattopadhyay, G.N., Morgan, T.W., and Raghuram, A., "An innovative technique for backbone network design", *IEEE Transactions on Systems,* Vol: 19, No:5 (1989) 1122-1132

[6] Cheng, S.T., "Topological optimization of a reliable communication network", *IEEE Transactions on Reliability,* Vol 47, No:3 (1998) 225-233.

[7] Chou, W., and Frank, H., "Routing strategies for computer network design", presented at Symp. Computer-Communication Networks and Teletraffic, Polytechnic Inst. of Brooklyn, NY, Apr. 4- (1972).

[8] Dutta, A., and Lim, J.I., “A multiperiod capacity planning model for backbone computer communication networks”, *Operations Research* 40 (1992) 689-705.

[9] Gallo, G., and Sodini, C., “Concave cost minimization on networks”, *European Journal of Operations Research* 3 (1979) 239-249.

[10] Gavish, B., “Topological design of computer communication networks-the overall design problem”, *European Journal of Operations Research* 58 (1992) 149-172.

[11] Gerla, M., and Kleinrock, L., “On the topological design of distributed computer networks”, *IEEE Trans. Comm.* COM-25 (1987) 48-60.

[12] Gersht, A., and Weihmayer, R., “Joint optimization of data network design and facility selection”, *IEEE Journal on Selected Areas in Communications* 8(9) (1990) 1667–1681.

[13] Kershenbaum, A., Kermani, P., and Grover, G.A, “MENTOR: An algorithm for mesh network topological optimization and routing*”, IEEE Transactions on Communications* 39(4) (1991) 503–513.

[14] Kennington,.L., “A survey of linear cost multicommodity network flows”, *Operations Research* 26/2 (1978) 209-236.

[15] Kleinrock, L., *Queuing Systems*, Vol. 2: *Computer Applications,* Wiley, New York, (1976).

[16] Lindberg, P., “Network structure optimization with respect to cost and dependability”, in Lada, L.,(ed) *Network Planning in the 1990’s,* Elsevier (1989) 373-378.

[17] Magnanti, T.L., and Wong, R.T., "Network design and transportation planning : models and algorithms", *Transportation Science* 18/1 (1984) 1-55.

[18] Magnanti, T.L., Mirchandani, P., and Vachani, R., "Modeling and solving the two-facility capacitated network loading problem", *Operations Research* 43 (1995) 142-157.

[19] Minoux, M., "Multinots de cout minimal avec functions de cout concaves", *Annales des Telecommunications* 31 (1976) 77-92.

[20] Minoux, M., "Network synthesis and optimum network design problems: Models, solution methods and applications", *Networks* 9 (1989) 313-360.

[21] Monma, C.L., and Sheng, D.D, "Backbone network design and analysis: A method for packet switching networks", *IEEE Journal on Selected Areas in Communications, Vol Sac-4 No:6* (1986) 946-965.

[22] Sharma, R.L., *Network topology optimization: The art and science of network design,* Van Nostrand Reinhol (1990)

[23] Stacey, C.H.E., Eyers, T., and Anido, G.,J., "A concave link elimination (CLE) procedure and lower bound for concave topology, capacity and flow assignment network design problems" *Telecommunication Systems,* Vol 13 (2000) 351-372

[24] Tanenbaum, A.S., *Computer Networks,* 4th Ed. Pearson Education (2003).

[25] Yaged, B.A., Jr. "Minimum cost routing for static network models", *Networks* 1, (1971) 139-172.

[26] Yaged, B.A., Jr. "Minimum cost routing for dynamic network models", *Networks* 3, (1973) 193-224.

[27] Zadeh, N., “On building minimum cost communication networks”, *Networks* 3, (1973) 315-331.

[28] Bazlamaçcı, C.F., “Optimised network design: minimum Spanning trees and minimum concave cost problems”, PhD Thesis, The University of Manchester Institute of Science and Technology, (1996)

[29] Ward, J.A, “Minimum aggregate concave cost multicommodity flows in strong-series-parallel networks”, *Mathematics of Operational Research* 24, (1999) 106-129.

[30] Yan, S., and Luo, S.C., “Probabilistic local search algorithms for concave cost transportation network problems”, *European Journal of Operational Research* 117, (1999) 511-521.

[31] Bazlamaçcı, C.F., “Disaggregate local search”, Unpublished Technical Report, (1998).

APPENDIX

COMPUTATIONAL RESULTS

The network classes in this work are represented in the form of *"N L P T"*, where

- N is the number of nodes in the network
- L is the percentile of the possible traffic requirement
- P is the concavity value
- T is the traffic distribution to nodes, Balanced or Unbalanced

Table A.1 Computational Results for various pricing curves in Yaged's Linearization Method

Network Type	$\partial\Phi / \partial\psi$	$\Phi(\psi)/\psi$	$\frac{(\Phi(\psi)+K)}{\psi}$	$\frac{(\Phi(\psi)+2K)}{\psi}$	$\frac{(\Phi(\psi)+Kf)}{\psi}$
25 10 0.3 B	10,81 ± 2,34	2,29 ± 1,04	1,70 ± 0,93	**1,37 ± 0,80**	2,57 ± 1,17
25 10 0.3 UB	10,44 ± 2,44	3,37 ± 1,40	1,88 ± 0,85	**1,64 ± 0,81**	4,05 ± 1,65
25 10 0.7 B	1,56 ± 0,34	0,97 ± 0,27	0,53 ± 0,22	**0,32 ± 0,19**	0,92 ± 0,26
25 10 0.7 UB	1,82 ± 0,43	1,23 ± 0,35	0,75 ± 0,26	**0,35 ± 0,19**	1,21 ± 0,34
25 10 V B	**0,02 ± 0,07**	23,03 ± 3,94	39,15 ± 5,72	51,14 ± 6,5	25,31 ± 4,24
25 10 V UB	**0,05 ± 0,12**	18,88 ± 3,73	34,45 ± 5,60	45,05 ± 6,85	20,78 ± 4,26
25 25 0.3 B	19,45 ± 3,65	5,03 ± 1,72	3,56 ± 1,50	**2,46 ± 1,27**	5,99 ± 1,84
25 25 0.3 UB	18,54 ± 3,19	4,34 ± 1,56	2,84 ± 1,20	**2,11 ± 1,04**	5,16 ± 1,57
25 25 0.7 B	2,90 ± 0,56	1,97 ± 0,43	1,27 ± 0,32	**0,65 ± 0,27**	1,88 ± 0,42
25 25 0.7 UB	2,89 ± 0,56	1,79 ± 0,42	1,11 ± 0,37	**0,57 ± 0,29**	1,79 ± 0,42
25 25 V B	**0,03 ± 0,05**	27,14 ± 4,41	48,28 ± 6,49	66,77 ± 8,50	29,25 ± 4,79
25 25 V UB	**0,03 ± 0,07**	27,11 ± 4,90	44,08 ± 6,55	61,41 ± 8,34	28,84 ± 4,99
25 50 0.3 B	15,10 ± 2,35	2,37 ± 0,77	1,49 ± 0,62	**1,27 ± 0,66**	2,61 ± 0,83
25 50 0.3 UB	13,64 ± 2,05	2,17 ± 0,75	1,30 ± 0,69	**1,09 ± 0,55**	2,67 ± 0,89
25 50 0.7 B	3,30 ± 0,46	1,78 ± 0,35	0,80 ± 0,23	**0,34 ± 0,18**	1,65 ± 0,34
25 50 0.7 UB	3,24 ± 0,45	1,85 ± 0,34	0,89 ± 0,24	**0,32 ±0,17**	1,72 ± 0,33

Table A.1 (Cont'd) Computational Results for various pricing curves in Yaged's Linearization Method

Network Type	$\partial\Phi / \partial\psi$	$\Phi(\psi) / \psi$	$\frac{(\Phi(\psi)+K)}{\psi}$	$\frac{(\Phi(\psi)+2K)}{\psi}$	$\frac{(\Phi(\psi)+Kf)}{\psi}$
25 50 V B	**0,04 ± 0,09**	27,64 ± 4,44	50,61 ± 6,90	69,95 ± 9,03	29,97 ± 4,78
25 50 V UB	**0,05 ± 0,09**	28,28 ± 4,51	46,53 ± 5,89	66,62 ± 7,93	30,12 ± 4,63
25 100 0.3	11,67 ± 1,89	1,86 ± 0,65	0,96 ± 0,41	**0,75 ± 0,44**	2,02 ± 0,56
25 100 0.7	2,65 ± 0,31	1,27 ± 0,24	0,62 ± 0,18	**0,14 ± 0,09**	1,19 ± 0,23
25 100 V	**0,00 ± 0,00**	30,26 ± 4,60	52,20 ± 7,51	69,98 ± 9,18	32,97 ± 4,99
50 10 0.3 B	34,14 ± 4,08	4,84 ± 1,38	2,42 ± 0,91	**1,98 ± 0,92**	6,30 ± 1,61
50 10 0.3 UB	29,23 ± 3,75	3,71 ± 1,12	1,95 ± 0,64	**1,46 ± 0,64**	5,10 ± 1,40
50 10 0.7 B	3,79 ± 0,48	2,04 ± 0,36	0,93 ± 0,23	**0,16 ± 0,11**	1,89 ± 0,34
50 10 0.7 UB	4,46 ± 0,53	2,59 ± 0,42	1,23 ± 0,30	**0,14 ± 0,10**	2,43 ± 0,41
50 10 V B	**0,00 ± 0,00**	27,90 ± 3,07	51,64 ± 4,78	72,52 ± 6,12	31,26 ± 3,19
50 10 V UB	**0,00 ± 0,00**	25,40 ± 2,86	47,19 ± 4,44	64,93 ± 5,50	27,69 ± 3,06
50 25 0.3 B	16,89 ± 1,56	1,70 ± 0,41	0,83 ± 0,28	**0,62 ± 0,26**	2,70 ± 0,56
50 25 0.3 UB	15,02 ± 1,38	1,69 ± 0,44	0,94 ± 0,30	**0,63 ± 0,27**	2,42 ± 0,52
50 25 0.7 B	5,74 ± 0,49	2,61 ± 0,35	1,03 ± 0,25	**0,20 ± 0,12**	2,37 ± 0,34
50 25 0.7 UB	5,81 ± 0,48	2,52 ± 0,37	0,99 ± 0,24	**0,16 ± 0,10**	2,27 ± 0,35
50 25 V B	**0,01 ± 0,06**	25,78 ± 2,80	47,21 ± 4,30	66,27 ± 5,83	28,94 ± 3,09
50 25 V UB	**0,00 ± 0,00**	27,81 ± 3,11	48,21 ± 5,03	69,28 ± 6,46	30,25 ± 3,19
50 50 0.3 B	14,24 ± 1,20	1,87 ± 0,40	0,70 ± 0,24	**0,41 ± 0,17**	2,65 ± 0,49
50 50 0.3 UB	12,92 ± 1,19	1,62 ± 0,37	0,62 ± 0,24	**0,42 ± 0,24**	2,54 ± 0,49
50 50 0.7 B	4,33 ± 0,34	1,80 ± 0,24	0,55 ± 0,13	**0,19 ± 0,10**	1,58 ± 0,23
50 50 0.7 UB	3,88 ± 0,31	1,54 ± 0,22	0,57 ± 0,13	**0,17 ± 0,09**	1,33 ± 0,21
50 50 V B	**0,00 ± 0,00**	26,12 ± 2,96	48,01 ± 4,19	68,32 ± 5,80	28,70 ± 3,07
50 50 V UB	**0,00 ± 0,00**	28,53 ± 2,96	51,25 ± 5,11	73,60 ± 6,55	31,31 ± 3,36
50 100 0.3	13,22 ± 1,00	1,70 ± 0,39	0,68 ± 0,23	**0,28 ± 0,16**	2,51 ± 0,46
50 100 0.7	2,68 ± 0,23	1,06 ± 0,15	0,38 ± 0,11	**0,08 ± 0,05**	1,00 ± 0,16
50 100 V UB	**0,00 ± 0,00**	29,19 ± 3,46	52,64 ± 5,19	71,74 ± 6,50	31,73 ± 3,76
75 10 0.3 UB	21,85 ± 2,62	1,90 ± 0,68	1,14 ± 0,67	**0,83 ± 0,56**	2,99 ± 0,90
75 10 0.7 B	6,55 ± 0,92	3,36 ± 0,73	1,36 ± 0,45	**0,09 ± 0,12**	3,19 ± 0,72
75 10 0.7 UB	7,44 ± 0,87	3,52 ± 0,59	1,36 ± 0,40	**0,09 ± 0,13**	3,32 ± 0,59
75 10 V B	**0,00 ± 0,00**	24,02 ± 3,59	45,74 ± 5,06	67,09 ± 6,90	27,17 ± 3,56
75 10 V UB	**0,00 ± 0,00**	24,77 ± 3,27	45,87 ± 4,99	65,82 ± 7,51	27,69 ± 3,29
75 25 0.3 B	16,20 ± 1,43	1,72 ± 0,50	0,54 ± 0,30	**0,23 ± 0,18**	2,79 ± 0,67
75 25 0.3 UB	14,96 ± 1,44	1,41 ± 0,49	0,74 ± 0,36	**0,32 ± 0,31**	2,42 ± 0,58
75 25 0.7 B	6,99 ± 0,63	2,72 ± 0,54	0,79 ± 0,54	**0,16 ± 0,13**	2,36 ± 0,51
75 25 0.7 UB	6,29 ± 0,64	2,28 ± 0,39	0,62 ± 0,22	**0,11 ± 0,12**	1,98 ± 0,40
75 25 V B	**0,00 ± 0,00**	23,75 ± 3,41	45,24 ± 5,35	63,69 ± 6,38	27,12 ± 3,78
75 25 V UB	**0,00 ± 0,00**	28,16 ± 4,13	49,58 ± 5,91	71,78 ± 8,00	31,82 ± 4,67
75 50 0.3 B	15,67 ± 1,52	1,57 ± 0,45	0,60 ± 0,32	**0,13 ± 0,11**	2,66 ± 0,55
75 50 0.3 UB	14,90 ± 1,41	1,53 ± 0,57	0,47 ± 0,26	**0,18 ± 0,18**	2,54 ± 0,77
75 50 0.7 B	4,04 ± 0,52	1,32 ± 0,28	0,40 ± 0,16	**0,20 ± 0,14**	1,14 ± 0,29
75 50 0.7 UB	3,82 ± 0,47	1,19 ± 0,28	0,37 ± 0,17	**0,22 ± 0,15**	1,01 ± 0,26
75 50 V B	**0,00 ± 0,00**	26,12 ± 3,41	48,24 ± 4,64	67,60 ± 6,63	29,01 ± 3,67
75 50 V UB	**0,00 ± 0,00**	29,00 ± 3,63	51,21 ± 5,54	68,90 ± 6,86	32,02 ± 3,86
75 100 0.3	14,77 ± 1,35	1,69 ± 0,45	0,48 ± 0,27	**0,16 ± 0,19**	2,45 ± 0,54
75 100 0.7	2,42 ± 0,29	0,86 ± 0,21	0,29 ± 0,12	**0,14 ± 0,10**	0,76 ± 0,20
75 100 V	0,00 ± 0,00	**25,91 ± 3,37**	**47,88 ± 5,57**	**68,93 ± 7,16**	29,29 ± 3,64

Table A.2 Computational Results of modification method for Minoux greedy

Network Type	Flow Reroute		Cycle Remove	
	Quality	Time	Quality	Time
25 10 0.3 B	0,35 ± 0,54	38	**0,05 ± 0,19**	**40**
25 10 0.3 UB	0,34 ± 0,50	34	**0,03 ± 0,09**	**36**
25 10 0.7 B	0,06 ± 0,25	31	0,05 ± 0,14	28
25 10 0.7 UB	**0,00 ± 0,00**	**29**	0,01 ± 0,04	24
25 10 V B	0,57 ± 1,16	46	**0,02 ± 0,06**	45
25 10 V UB	0,13 ± 0,33	37	**0,08 ± 0,29**	39
25 25 0.3 B	0,65 ± 0,84	193	**0,10 ± 0,20**	192
25 25 0.3 UB	0,60 ± 0,65	190	**0,26 ± 0,40**	188
25 25 0.7 B	**0,00 ± 0,00**	**173**	0,03 ± 0,06	173
25 25 0.7 UB	**0,00 ± 0,01**	**152**	0,11 ± 0,13	158
25 25 V B	0,14 ± 0,54	212	**0,03 ± 0,13**	**211**
25 25 V UB	0,39 ± 1,05	191	**0,08 ± 0,29**	**193**
25 50 0.3 B	0,25 ± 0,39	433	**0,18 ± 0,29**	**431**
25 50 0.3 UB	0,25 ± 0,34	403	**0,08 ± 0,25**	**401**
25 50 0.7 B	**0,10 ± 0,25**	**387**	0,11 ± 0,27	402
25 50 0.7 UB	**0,03 ± 0,07**	**352**	0,08 ± 0,09	350
25 50 V B	**0,00 ± 0,00**	**448**	0,04 ± 0,15	445
25 50 V UB	**0,06 ± 0,14**	**412**	0,09 ± 0,26	407
25 100 0.3	**0,11 ± 0,24**	**815**	0,19 ± 0,46	776
25 100 0.7	**0,01 ± 0,03**	**699**	0,04 ± 0,04	698
25 100 V	0,02 ± 0,07	757	**0,14 ± 0,47**	**743**
50 10 0.3 B	0,84 ± 0,71	1979	**0,05 ± 0,11**	**2079**
50 10 0.3 UB	0,82 ± 0,77	1808	**0,10 ± 0,18**	**1886**
50 10 0.7 B	**0,01 ± 0,02**	**1577**	0,06 ± 0,06	1583
50 10 0.7 UB	**0,01 ± 0,02**	**1456**	0,09 ± 0,11	1453
50 10 V B	0,29 ± 0,51	2008	**0,01 ± 0,02**	**2045**
50 10 V UB	0,41 ± 0,44	1888	**0,02 ± 0,05**	**1916**
50 25 0.3 B	0,76 ± 0,55	6794	**0,03 ± 0,10**	**7015**
50 25 0.3 UB	0,66 ± 0,63	6206	**0,14 ± 0,23**	**6475**
50 25 0.7 B	**0,01 ± 0,03**	**6106**	0,14 ± 0,14	6144
50 25 0.7 UB	**0,01 ± 0,02**	**5908**	0,18 ± 0,11	5775
50 25 V B	0,43 ± 0,46	14089	**0,04 ± 0,10**	**7071**
50 25 V UB	0,34 ± 0,37	12991	**0,04 ± 0,12**	**6532**
50 0.3 B	0,38 ± 0,33	27511	**0,12 ± 0,21**	**13851**
50 0.3 UB	0,41 ± 0,44	24371	**0,35 ± 0,48**	**12455**
50 0.7 B	**0,01 ± 0,05**	**23951**	0,09 ± 0,06	12218
50 0.7 UB	**0,00 ± 0,01**	**23991**	0,08 ± 0,05	12047
50 V B	0,32 ± 0,45	27813	**0,07 ± 0,12**	**13983**
50 V UB	0,25 ± 0,36	25431	**0,06 ± 0,12**	**12633**
50 100 0.3	0,22 ± 0,33	49423	**0,19 ± 0,21**	**24792**
50 100 0.7	**0,00 ± 0,01**	**91153**	0,06 ± 0,04	43139
50 100 V UB	0,09 ± 0,13	142883	**0,00 ± 0,01**	**45452**

Table A.3 Computational Results for evaluation of Disaggregate Local Search.

HC represents the result associated to starting solution with Hop Count. Similarly, PP represents Physical path distance.

Network Type	Disaggregate Largest		Disaggregate Smallest		Disaggregate Random		Disaggregate Improvement	
	HC	PP	HC	PP	HC	PP	HC	PP
25 10 0.3 B	23,62 ± 3,28	5,50 ± 1,72	22,72 ± 2,74	6,18 ± 1,89	22,76 ± 3,00	6,13 ± 1,89	15,72 ± 0,82	**0,98 ± 0,89**
25 10 0.3 UB	18,30 ± 3,36	5,39 ± 1,95	18,48 ± 3,47	5,48 ± 1,89	18,24 ± 3,36	4,69 ± 1,58	10,61 ± 7,23	**2,50 ± 1,27**
25 10 0.7 B	2,33 ± 1,89	1,12 ± 1,10	2,26 ± 0,79	1,85 ± 0,61	2,16 ± 1,84	1,16 ± 0,48	2,45 ± 0,81	**0,93 ± 0,44**
25 10 0.7 UB	2,11 ± 0,92	1,42 ± 0,61	2,84 ± 0,88	1,75 ± 0,57	2,19 ± 0,76	1,26 ± 0,54	2,85 ± 0,86	**0,94 ± 0,41**
n25 10 V B	**6,04 ± 2,58**	11,34 ± 3,82	11,68 ± 3,81	13,72 ± 3,72	7,80 ± 3,10	10,97 ± 3,57	16,38 ± 3,99	14,84 ± 3,97
n25 10 V UB	**7,34 ± 3,11**	10,42 ± 3,10	12,15 ± 3,79	13,08 ± 3,94	8,69 ± 3,23	10,54 ± 3,41	16,61 ± 4,01	11,43 ± 3,34
n25 25 0.3 B	22,83 ± 3,44	4,30 ± 1,65	24,28 ± 3,97	4,48 ± 1,67	20,56 ± 2,98	**2,09 ± 1,08**	18,22 ± 3,10	3,57 ± 1,82
n25 25 0.3 UB	20,23 ± 3,33	3,76 ± 1,46	20,95 ± 3,71	4,40 ± 1,46	18,30 ± 3,09	**1,98 ± 0,96**	15,10 ± 3,14	3,85 ± 1,45
n25 25 0.7 B	3,92 ± 1,19	1,31 ± 0,45	3,71 ± 1,16	1,50 ± 0,55	2,39 ± 0,79	**0,77 ± 0,39**	4,65 ± 1,13	1,42 ± 0,54
25 25 0.7 UB	3,97 ± 1,27	1,26 ± 0,44	4,35 ± 1,25	1,56 ± 0,58	3,15 ± 0,86	**0,81 ± 0,40**	4,94 ± 1,27	1,38 ± 0,48
25 25 V B	18,71 ± 5,00	**4,27 ± 2,14**	33,40 ± 6,52	8,08 ± 2,97	21,05 ± 5,38	4,30 ± 2,17	21,09 ± 4,99	4,80 ± 2,36
25 25 V UB	14,95 ± 4,84	**5,67 ± 2,87**	24,21 ± 5,70	10,37 ± 3,70	15,00 ± 5,12	5,99 ± 2,75	16,78 ± 4,88	6,72 ± 3,01
25 50 0.3 B	12,41 ± 2,67	4,09 ± 1,51	13,34 ± 2,55	3,92 ± 1,66	8,74 ± 2,28	**1,35 ± 0,84**	11,17 ± 2,77	3,46 ± 1,44
25 50 0.3 UB	10,28 ± 2,57	4,12 ± 1,56	10,54 ± 2,54	3,86 ± 1,59	6,76 ± 2,08	**1,34 ± 0,86**	8,58 ± 2,38	4,22 ± 1,70
25 50 0.7 B	3,73 ± 0,96	0,97 ± 0,38	5,21 ± 1,07	1,09 ± 0,35	2,50 ± 0,74	**0,26 ± 0,17**	4,82 ± 1,15	1,16 ± 0,34
25 50 0.7 UB	3,81 ± 1,05	0,93 ± 0,34	5,68 ± 1,47	0,92 ± 0,32	2,22 ± 0,73	**0,25 ± 0,20**	4,61 ± 1,25	1,06 ± 0,37

Table A.3 (Cont'd) Computational Results for evaluation of Disaggregate Local Search

Network Type	Disaggregate Largest		Disaggregate Smallest		Disaggregate Random		Disaggregate Improvement	
	HC	PP	HC	PP	HC	PP	HC	PP
25 50 V B	7,67 ± 3,17	10,50 ± 3,71	9,19 ± 2,95	13,09 ± 4,81	**5,28 ± 2,44**	8,76 ± 3,64	5,47 ± 2,64	9,03 ± 3,78
25 50 V UB	7,50 ± 2,45	10,99 ± 4,31	10,26 ± 3,58	11,57 ± 4,79	**4,97 ± 2,48**	8,28 ± 4,31	5,58 ± 2,77	10,06 ± 4,85
25 100 0.3	7,63 ± 1,86	4,47 ± 1,42	6,99 ± 1,67	4,54 ± 1,47	1,95 ± 0,85	**1,62 ± 0,83**	5,51 ± 1,51	4,61 ± 1,85
25 100 0.7	1,43 ± 0,49	1,25 ± 0,34	2,44 ± 0,68	1,26 ± 0,35	**0,24 ± 0,18**	0,64 ± 0,23	1,35 ± 0,45	1,55 ± 0,35
25 100 V	3,88 ± 1,26	21,32 ± 5,40	5,18 ± 1,45	22,94 ± 5,74	**0,64 ± 0,50**	19,10 ± 5,05	2,79 ± 1,01	20,98 ± 5,18
50 10 0.3 B	31,18 ± ,43	6,52 ± 1,71	31,58 ± 3,14	4,96 ± 1,20	25,22 ± 2,28	**0,49 ± 0,42**	24,45 ± 2,65	4,36 ± 1,59
50 10 0.3 UB	26,21 ± 2,52	5,11 ± 1,62	26,61 ± 2,52	5,05 ± 1,42	21,68 ± 2,25	**0,79 ± 0,53**	18,44 ± 2,21	3,60 ± 1,46
50 10 0.7 B	3,57 ± 0,78	1,63 ± 0,45	3,59 ± 0,64	2,04 ± 0,48	1,31 ± 0,41	**0,23 ± 0,16**	4,12 ± 0,75	1,34 ± 0,36
50 10 0.7 UB	3,83 ± 0,77	1,60 ± 0,46	3,80 ± 0,73	1,85 ± 0,47	1,10 ± 0,44	**0,34 ± 0,20**	4,58 ± 0,80	1,66 ± 0,41
50 10 V B	23,57 ± 4,05	4,48 ± 1,22	43,75 ± 5,01	6,91 ± 1,78	22,78 ± 3,74	**0,35 ± 0,39**	32,43 ± 4,69	3,50 ± 1,24
50 10 V UB	17,88 ± 3,63	4,36 ± 1,46	31,40 ± 3,96	7,51 ± 1,76	16,21 ± 2,91	**0,32 ± 0,34**	25,57 ± 3,18	5,24 ± 1,32
50 25 0.3 B	17,13 ± 2,79	4,42 ± 1,46	17,83 ± 3,20	3,42 ± 1,26	12,32 ± 2,57	**0,26 ± 0,28**	16,64 ± 3,10	3,58 ± 1,46
50 25 0.3 UB	15,30 ± 3,14	3,79 ± 1,54	15,24 ± 3,12	3,35 ± 1,31	11,05 ± 2,55	**0,13 ± 0,26**	13,98 ± 3,24	3,65 ± 1,29
50 25 0.7 B	6,04 ± 1,31	1,16 ± 0,48	6,82 ± 1,44	1,17 ± 0,44	3,03 ± 0,98	**0,07 ± 0,10**	6,35 ± 1,38	1,44 ± 0,48
50 25 0.7 UB	5,11 ± 1,11	1,52 ± 0,47	7,26 ± 1,46	1,21 ± 0,40	3,04 ± 0,78	**0,09 ± 0,14**	6,43 ± 1,33	1,38 ± 0,50
50 25 V B	21,64 ± 4,44	2,75 ± 1,25	23,00 ± 4,72	3,83 ± 1,90	18,31 ± 4,27	**0,06 ± 0,20**	20,30 ± 4,53	2,23 ± 1,12

Table A.3 (Cont'd) Computational Results for evaluation of Disaggregate Local Search

Network Type	Disaggregate Largest		Disaggregate Smallest		Disaggregate Random		Disaggregate Improvement	
	HC	PP	HC	PP	HC	PP	HC	PP
50 25 V UB	20,42 ± 4,86	2,29 ± 0,87	23,92 ± 4,55	4,81 ± 1,77	16,24 ± 4,76	**0,10 ± 0,16**	18,67 ± 4,60	2,74 ± 0,94
50 50 0.3 B	10,47 ± 3,51	3,08 ± 1,99	9,74 ± 3,27	3,23 ± 1,93	6,02 ± 2,68	**0,23 ± 0,43**	10,61 ± 3,20	4,69 ± 2,61
50 50 0.3 UB	8,99 ± 2,73	3,59 ± 1,72	7,92 ± 3,28	3,28 ± 1,74	4,11 ± 2,22	**0,28 ± 0,58**	8,57 ± 3,11	2,78 ± 1,79
50 50 0.7 B	5,95 ± 1,96	0,75 ± 0,48	9,11 ± 2,97	0,75 ± 0,51	3,95 ± 1,39	**0,11 ± 0,28**	6,49 ± 2,25	1,03 ± 0,58
50 50 V B	8,93 ± 4,27	3,14 ± 2,64	9,56 ± 4,50	4,95 ± 4,63	5,26 ± 3,20	**0,83 ± 1,91**	8,66 ± 4,09	3,47 ± 2,25
50 50 V UB	4,36 ± 2,67	3,65 ± 1,69	5,48 ± 2,81	5,11 ± 2,93	**1,45 ± 1,90**	**1,56 ± 1,36**	3,53 ± 2,56	3,46 ± 1,99
50 100 0.3	6,98 ± 3,16	3,68 ± 1,54	5,90 ± 3,21	3,86 ± 2,18	**1,29 ± 1,09**	**1,08 ± 1,36**	6,09 ± 2,74	3,50 ± 2,65
50 100 0.7	1,47 ± 1,30	0,76 ± 0,52	2,26 ± 1,14	0,99 ± 0,60	**0,25 ± 0,27**	0,29 ± 0,41	1,19 ± 1,18	1,02 ± 0,64
50 100 V UB	4,90 ± 2,74	13,32 ± 6,28	4,01 ± 1,76	13,19 ± 6,46	**0,25 ± 0,56**	10,56 ± 5,68	2,75 ± 2,30	12,69 ± 6,22
75 10 0.3 B	23,69 ± 2,06	4,20 ± 1,27	24,10 ± 2,17	3,93 ± 0,99	19,39 ± 1,84	**0,21 ± 0,23**	19,79 ± 1,82	3,83 ± 1,19
n75 10 0.3 UB	22,48 ± 2,03	4,01 ± 0,94	22,77 ± 2,11	4,11 ± 0,97	18,59 ± 1,69	**0,30 ± 0,28**	17,42 ± 1,81	3,13 ± 0,89
n75 10 0.7 B	4,32 ± 0,65	1,53 ± 0,41	4,00 ± 0,70	1,60 ± 0,34	1,91 ± 0,43	**0,13 ± 0,13**	4,80 ± 0,70	1,52 ± 0,32
n75 10 0.7 UB	4,76 ± 0,73	1,39 ± 0,36	5,09 ± 0,80	1,64 ± 0,40	2,44 ± 0,50	**0,09 ± 0,10**	5,35 ± 0,81	1,28 ± 0,32
n75 10 V B	36,85 ± 3,86	2,83 ± 0,81	47,93 ± 4,69	4,23 ± 1,04	34,27 ± 3,59	**0,20 ± 0,20**	37,93 ± 4,07	2,51 ± 0,91
n75 10 V UB	29,73 ± 3,44	3,30 ± 0,99	39,39 ± 4,16	4,87 ± 1,13	26,93 ± 8,61	**0,19 ± 0,20**	32,16 ± 3,68	3,12 ± 0,94
75 25 0.3 B	13,33 ± 4,29	3,08 ± 1,74	12,82 ± 3,00	2,06 ± 1,62	9,34 ± 3,17	**0,33 ± 0,44**	13,46 ± 5,79	2,93 ± 2,20

Table A.3 (Cont'd) Computational Results for evaluation of Disaggregate Local Search

Network Type	Disaggregate Largest		Disaggregate Smallest		Disaggregate Random		Disaggregate Improvement	
	HC	**PP**	**HC**	**PP**	**HC**	**PP**	**HC**	**PP**
75 25 0.3 UB	12,54 ± 3,00	3,38 ± 2,37	12,51 ± 3,18	2,92 ± 1,49	8,31 ± 2,55	**0,11 ± 0,37**	11,54 ± 4,18	2,33 ± 2,26
75 25 0.7 B	6,57 ± 2,30	1,38 ± 0,55	8,49 ± 2,65	1,21 ± 0,90	4,55 ± 1,61	**0,02 ± 0,06**	6,84 ± 2,51	1,17 ± 0,72
75 25 0.7 UB	8,59 ± 2,14	1,37 ± 0,96	9,01 ± 2,55	1,52 ± 0,97	4,29 ± 2,07	**0,04 ± 0,10**	8,06 ± 2,83	1,35 ± 1,25
50 50 0.7 UB	4,14 ± 1,94	0,82 ± 0,47	6,49 ± 1,95	0,83 ± 0,37	2,57 ± 1,13	**0,02 ± 0,05**	4,74 ± 1,80	0,97 ± 0,43
75 25 V B	18,27 ± 5,16	1,78 ± 1,21	19,07 ± 5,29	2,52 ± 1,07	15,62 ± 4,51	**0,04 ± 0,14**	17,48 ± 4,93	1,51 ± 1,03
75 25 V UB	20,14 ± 3,96	2,48 ± 1,74	20,95 ± 4,00	2,81 ± 1,48	17,15 ± 3,77	**0,00 ± 0,00**	17,89 ± 6,27	2,97 ± 2,40
75 50 0.3 B	8,22 ± 2,45	2,83 ± 1,60	7,66 ± 2,74	3,07 ± 1,76	5,09 ± 2,32	**0,20 ± 0,52**	8,70 ± 4,13	2,62 ± 2,02
75 50 0.3 UB	8,72 ± 3,49	2,95 ± 1,27	8,00 ± 3,17	3,19 ± 1,04	5,29 ± 2,79	**0,00 ± 0,00**	7,78 ± 5,10	2,15 ± 2,01
75 50 0.7 B	6,91 ± 2,63	0,97 ± 0,59	11,36 ± 1,99	0,95 ± 0,56	5,69 ± 1,92	**0,08 ± 0,19**	7,75 ± 3,47	0,60 ± 0,75
75 50 0.7 UB	4,47 ± 1,57	0,93 ± 0,50	7,11 ± 1,77	1,14 ± 0,65	3,52 ± 1,37	**0,01 ± 0,02**	5,07 ± 2,27	1,17 ± 0,74
75 50 V B	5,99 ± 3,05	2,38 ± 1,54	6,50 ± 3,40	2,60 ± 1,71	3,84 ± 2,93	**0,16 ± 0,47**	6,53 ± 5,81	1,42 ± 1,64
75 50 V UB	6,79 ± 2,55	2,82 ± 1,96	6,77 ± 3,30	2,62 ± 1,35	3,59 ± 2,15	**0,44 ± 0,72**	6,44 ± 2,49	2,78 ± 2,22
75 100 0.3	6,17 ± 2,26	2,55 ± 1,71	5,30 ± 2,66	2,84 ± 1,31	1,76 ± 1,39	**0,25 ± 0,50**	Not Tested	Not Tested
75 100 0.7	1,35 ± 1,07	0,82 ± 0,61	1,73 ± 0,92	0,79 ± 0,57	**0,31 ± 0,58**	0,56 ± 1,01	Not Tested	Not Tested
75 100 V	2,77 ± 1,62	5,97 ± 1,78	2,66 ± 1,63	5,70 ± 2,14	**0,12 ± 0,32**	3,90 ± 2,83	Not Tested	Not Tested

Table A.4 Quality comparison of circuit switching methods.

Y: Yaged's Linearization **MG**: Minoux Greedy **MAG**: Minoux Accelerated Greedy **MMG**: Modified Minoux Greedy **MAMG**: Modified Accelerated Minoux Greedy **DLS**: Disaggregate Local Search

Network Type	Y	MG	MAG	MMG	MAMG	DLS
25 10 0.3 B	37,55 ± 3,27	1,14 ± 0,27	1,14 ± 0,27	0,22 ± 0,2	**0,22 ± 0,2**	6,77 ± 1,14
25 10 0.3 UB	32,00 ± 3,12	2,59 ± 0,71	2,59 ± 0,71	1,38 ± 0,66	**1,38 ± 0,66**	3,76 ± 1,11
25 10 0.7 B	7,35 ± 0,74	4,83 ± 0,57	5,05 ± 0,58	4,43 ± 0,57	4,77 ± 0,59	**0,007 ± 0,02**
25 10 0.7 UB	7,50 ± 0,81	5,33 ± 0,61	5,62 ± 0,62	4,89 ± 0,62	5,32 ± 0,63	**0,006 ± 0,02**
25 10 V B	76,6 ± 11,33	75,02 ± 11,06	75,23 ± 11,09	71,96 ± 10,90	72,20 ± 10,95	**0,05 ± 0,24**
25 10 V UB	88,73 ± 12,66	87,82 ± 12,44	87,91 ± 12,46	84,22 ± 12,20	84,35 ± 12,22	**0,007 ± 0,03**
25 25 0.3 B	23,65 ± 2,93	2,27 ± 0,22	2,27 ± 0,22	0,0006 ± 0,002	**0,0006 ± 0,002**	20,69 ± 1,42
25 25 0.3 UB	22,49 ± 2,59	2,28 ± 0,24	2,28 ± 0,24	0,0001 ± 0,004	**0,0001 ± 0,004**	20,46 ± 1,32
25 25 0.7 B	7,50 ± 0,80	4,71 ± 0,53	5,41 ± 0,56	4,08 ± 0,53	5,05 ± 0,56	**0,01 ± 0,03**
25 25 0.7 UB	6,78 ± 0,76	5,01 ± 0,54	5,06 ± 0,56	4,32 ± 0,52	5,18 ± 0,54	**0,02 ± 0,04**
25 25 V B	14,70 ± 4,44	12,33 ± 4,28	12,35 ± 4,29	8,95 ± 4,07	8,96 ± 4,07	**2,7 ± 0,99**
25 25 V UB	17,92 ± 4,56	15,62 ± 4,51	15,63 ± 4,51	11,97 ± 4,37	11,97 ± 4,37	**2,09 ± 0,92**
25 50 0.3 B	13,29 ± 1,21	2,81 ± 0,25	2,81 ± 0,25	0,00 ± 0,00	**0,00 ± 0,00**	19,25 ± 1,27
25 50 0.3 UB	14,06 ± 1,26	2,86 ± 0,25	2,86 ± 0,25	0,00 ± 0,00	**0,00 ± 0,00**	19,63 ± 1,32
25 50 0.7 B	2,92 ± 0,52	4,74 ± 0,50	5,44 ± 0,52	4,12 ± 0,50	5,08 ± 0,52	**0,02 ± 0,03**
25 50 0.7 UB	2,72 ± 0,47	5,08 ± 0,50	5,85 ± 0,51	4,36 ± 0,51	5,44 ± 0,51	**0,02 ± 0,03**
25 50 V B	14,85 ± 3,46	11,80 ± 3,27	11,81 ± 3,28	7,88 ± 2,93	7,87 ± 2,93	**4,64 ± 1,68**
25 50 V UB	16,42 ± 4,57	13,55 ± 4,25	13,59 ± 4,29	9,16 ± 3,88	9,16 ± 3,88	**4,30 ± 1,69**
25 100 0.3	10,07 ±1,05	3,02 ± 0,24	3,02 ± 0,24	0,00 ± 0,00	**0,00 ± 0,00**	16,04 ± 1,12
25 100 0.7	1,18 ± 0,25	5,29 ± 0,41	6,11 ± 0,45	4,64 ± 0,40	5,75 ± 0,45	**0,11 ± 0,07**
25 100 V	9,51 ± 3,56	5,25 ± 3,46	5,27 ± 3,46	1,32 ± 3,22	**1,32 ± 3,22**	3,81 ± 0,69
50 10 0.3 B	29,4 ± 2,46	2,85 ± 0,20	2,85 ± 0,20	0,00 ± 0,00	**0,00 ± 0,00**	22,53 ± 0,81
50 10 0.3 UB	29,8 ± 2,26	2,85 ± 0,21	2,85 ± 0,21	0,001 ± 0,004	**0,001 ± 0,004**	20,64 ± 0,88
50 10 0.7 B	10,28 ± 0,58	9,13 ± 0,43	9,87 ± 0,46	8,56 ± 0,43	9,56 ± 0,46	**0,00 ± 0,00**

Table A.4 (Cont'd) Quality comparison of circuit switching methods.

Network Type	Y	MG	MAG	MMG	MAMG	DLS
50 10 0.7 UB	9,54 ± 0,63	10,15 ± 0,51	10,92 ± 0,54	9,41 ± 0,51	10,49 ± 0,54	**0,00 ± 0,00**
50 10 V B	12,30 ± 2,16	9,03 ± 2,08	9,03 ± 2,08	6,22 ± 1,97	6,22 ± 1,97	**0,76 ± 0,33**
50 10 V UB	18,22 ± 2,7	15,71 ± 2,74	15,71 ± 2,74	12,74 ± 2,65	12,74 ± 2,65	**0,17 ± 0,18**
50 25 0.3 B	14,21 ± 0,89	4,07 ± 0,24	4,07 ± 0,24	0,00 ± 0,00	**0,00 ± 0,00**	23,60 ± 0,91
50 25 0.3 UB	15,98 ± 0,91	4,09 ± 0,24	4,09 ± 0,24	0,00 ± 0,00	**0,00 ± 0,00**	24,05 ± 0,91
50 25 0.7 B	3,35 ± 0,46	9,91 ± 0,48	10,88 ± 0,52	9,23 ± 0,47	10,51 ± 0,52	**0,01 ± 0,03**
50 25 0.7 UB	2,79 ± 0,37	10,83 ± 0,50	11,84 ± 0,53	10,83 ± 0,50	11,38 ± 0,52	**0,007 ± 0,02**
50 25 V B	8,37 ± 1,06	3,79 ± 0,78	3,79 ± 0,78	0,56 ± 0,74	**0,56 ± 0,74**	3,90 ± 0,54
50 25 V UB	9,04 ± 1,28	4,57 ± 1,03	4,57 ± 1,03	0,93 ± 0,95	**0,93 ± 0,95**	3,85 ± 0,59
50 50 0.3 B	11,24 ± 0,71	4,45 ± 0,25	4,45 ± 0,25	0,00 ± 0,00	**0,00 ± 0,00**	21,38 ± 0,82
50 50 0.3 UB	12,43 ± 0,83	4,53 ± 0,24	4,53 ± 0,24	0,00 ± 0,00	**0,00 ± 0,00**	21,69 ± 0,90
50 50 0.7 B	0,57 ± 0,18	10,28 ± 0,45	11,23 ± 0,48	9,66 ± 0,44	10,91 ± 0,47	**0,25 ± 0,11**
50 50 0.7 UB	0,74 ± 0,22	11,10 ± 0,51	12,08 ± 0,54	10,39 ± 0,48	11,69 ± 0,54	**0,15 ± 0,08**
50 50 V B	9,51 ± 0,97	3,55 ± 0,26	3,55 ± 0,26	0,05 ± 0,16	**0,05 ± 0,16**	5,77 ± 0,61
50 50 V UB	9,87 ± 0,99	4,30 ± 0,50	4,30 ± 0,50	0,42 ± 0,38	**0,42 ± 038**	7,70 ± 1,36
50 100 0.3	8,08 ± 0,65	4,64 ± 0,23	4,64 ± 0,23	0,00 ± 0,00	**0,00 ± 0,00**	18,44 ± 0,74
50 100 0.7	**0,23 ± 0,11**	10,44 ± 0,44	11,36 ± 0,47	9,88 ± 0,43	11,07 ± 0,46	0,38 ± 0,11
50 100 V UB	12,68 ± 1,14	5,73 ± 0,77	5,73 ± 0,77	1,83 ± 0,70	1,83 ± 0,70	**1,49 ± 0,51**
75 10 0.3 B	18,58 ± 1,61	4,27 ± 0,35	4,27 ± 0,35	0,00 ± 0,00	**0,00 ± 0,00**	27,97 ± 1,29
75 10 0.3 UB	19,96 ± 1,67	4,01 ± 0,32	4,01 ± 0,32	0,00 ± 0,00	**0,00 ± 0,00**	26,87 ± 1,18
75 10 0.7 B	8,32 ± 0,79	11,52 ± 0,56	12,48 ± 0,59	10,88 ± 0,56	12,16 ± 0,59	**0,00 ± 0,00**
75 10 0.7 UB	6,76 ± 0,76	13,07 ± 0,75	14,02 ± 0,77	12,29 ± 0,75	13,58 ± 0,77	**0,00 ± 0,00**
75 10 V B	8,58 ± 1,27	3,65 ± 0,96	3,65 ± 0,96	0,67 ± 0,91	**0,67 ± 0,91**	3,72 ± 0,99
75 10 V UB	8,34 ± 1,44	4,11 ± 0,93	4,11 ± 0,93	0,90 ± 0,89	**0,90 ± 0,89**	2,71 ± 0,91
75 25 0.3 B	12,98 ± 1,01	5,19 ± 0,37	5,19 ± 0,37	0,00 ± 0,00	**0,00 ± 0,00**	25,98 ± 1,14

Table A.4 (Cont'd) Quality comparison of circuit switching methods.

Network Type	Y	MG	MAG	MMG	MAMG	DLS
75 25 0.3 UB	14,56 ± 1,11	4,90 ± 0,34	4,90 ± 0,34	0,00 ±0,00	**0,00 ± 0,00**	26,00 ± 1,21
75 25 0.7 B	1,51 ± 0,45	13,28 ± 0,83	14,32 ± 0,88	12,70 ± 0,80	14,02 ± 0,87	**0,10 ± 0,14**
75 25 0.7 UB	1,44 ± 0,46	14, 99 ± 0,73	16,12 ± 0,77	14,21 ± 0,71	15,69 ± 0,76	**0,05 ± 0,06**
75 25 V B	10,35 ± 1,16	3,55 ± 0,25	3,55 ± 0,25	0,00 ± 0,00	**0,00 ± 0,00**	7,65 ± 0,82
75 25 V UB	10,49 ± 1,16	3,84 ± 0,36	3,84 ± 0,36	0,08 ± 0,23	**0,08 ± 0,23**	7,11 ± 0,92
75 50 0.3 B	9,47 ± 0,96	5,37 ± 0,40	5,37 ± 0,40	0,00 ± 0,00	**0,00 ± 0,00**	22,63 ± 1,19
75 50 0.3 UB	11,37 ± 1,09	5,44 ± 0,36	5,44 ± 0,36	0,00 ± 0,00	**0,00 ± 0,00**	23,88 ± 1,11
75 50 0.7 B	0,43 ± 0,25	15,04 ± 0,87	16,11 ± 0,90	14,46 ± 0,86	15,83 ± 0,90	**0,30 ± 0,17**
75 50 0.7 UB	0,50 ± 0,27	15,47 ± 0,81	16,58 ± 0,88	14,79 ± 0,79	16,23 ± 0,87	**0,28 ± 0,18**
75 50 V B	11,19 ± 1,09	3,71 ± 0,28	3,71 ± 0,28	0,00 ± 0,00	**0,00 ± 0,00**	8,45 ± 0,80
75 50 V UB	11,74 ± 1,15	4,01 ± 0,25	4,01 ± 0,25	0,00 ± 0,00	**0,00 ± 0,00**	8,98 ± 0,86
75 100 0.3	6,40 ± 0,82	5,72 ± 0,38	5,72 ± 0,38	0,00 ± 0,00	**0,00 ± 0,00**	19,31 ± 1,05
75 100 0.7	**0,23 ± 0,20**	15,01 ± 0,66	16,06 ± 0,68	15,01 ± 0,66	15,01 ± 0,68	**0,53 ± 0,20**
75 100 V	12,72 ± 1,29	4,19 ± 0,32	4,19 ± 0,32	0,07 ± 0,15	**0,07 ± 0,15**	5,26 ± 1,13

Table A.5 Average computational time (ms) requirements of circuit switching methods.

Y: Yaged's Linearization **MG**: Minoux Greedy **MAG**: Minoux Accelerated Greedy **MMG**: Modified Minoux Greedy **MAMG**: Modified Accelerated Minoux Greedy **DLS**: Disaggregate Local Search

Network Type	Y	MG	MAG	MMG	MAMG	DLS
25 10 0.3 B	6.4	43.7	34.13	42.06	**34.45**	112.5
25 10 0.3 UB	7.2	40.1	29.55	38.42	**30.48**	120.2
25 10 0.7 B	5,32	28,94	8,43	29,12	8,9	**89.92**
25 10 0.7 UB	5,28	28,66	8,2	29,06	8,9	**90,7**
25 10 V B	4,7	44,44	23,86	43,13	25,48	**18,69**
25 10 V UB	5,6	40,22	20,91	37,61	22,32	**18,96**
25 25 0.3 B	30	203	187	200	**190**	1272
25 25 0.3 UB	24	189	173	187	**176**	1238
25 25 0.7 B	24	169	56	170	56	**1352**
25 25 0.7 UB	26	154	50	154	51	**1289**
25 25 V B	17	210	164	206	170	**51**
25 25 V UB	18	188	144	184	151	**50**
25 50 0.3 B	59	437	413	428	**419**	2726
25 50 0.3 UB	55	404	380	396	**387**	2532
25 50 0.7 B	61	385	156	385	157	**3480**
25 50 0.7 UB	45	362	147	364	148	**3302**
25 50 V B	39	431	365	423	379	**2058**
25 50 V UB	40	410	346	401	362	**1918**
25 100 0.3	127	761	757	760	**754**	6326
25 100 0.7	151	682	322	684	323	**13620**
25 100 V	90	780	762	774	**736**	4504
50 10 0.3 B	161	3995	3813	3918	**3827**	18382
50 10 0.3 UB	164	4005	3472	3611	**3442**	17606
50 10 0.7 B	117	3057	765	3058	802	**18841**
50 10 0.7 UB	131	2838	690	2809	707	**19324**
50 10 V B	118	4117	4031	4004	3813	**12106**
50 10 V UB	103	3602	3126	3550	3182	**11759**
50 25 0.3 B	367	14229	13788	14306	**13687**	50837
50 25 0.3 UB	375	13069	11651	12998	**11094**	49571
50 25 0.7 B	540	12905	5011	12993	5145	**75989**
50 25 0.7 UB	520	11662	4532	11849	4697	**70897**
50 25 V B	335	14380	12673	14567	**12686**	36443
50 25 V UB	321	13675	12318	13559	**12348**	36153
50 50 0.3 B	786	28458	26290	27297	**26901**	96462
50 50 0.3 UB	734	26791	25887	26284	**26214**	90286
50 50 0.7 B	1321	24975	11754	24980	11965	**195744**
50 50 0.7 UB	1254	25336	11024	23900	11281	**182186**
50 50 V B	742	26796	25745	29045	**25947**	81813
50 50 V UB	710	24606	23775	24101	**23817**	42395
50 100 0.3	1592	45956	44162	52507	**46214**	171949
50 100 0.7	2894	42607	22727	40745	22780	**358346**
50 100 V UB	1600	47089	44832	49823	47001	**116202**
75 10 0.3 B	625	36611	36122	36534	**37738**	107301
75 10 0.3 UB	1123	32574	32442	35750	**32350**	98345
75 10 0.7 B	646	29144	8595	29776	8954	**133332**
75 10 0.7 UB	692	26214	8092	26311	7875	**134152**
75 10 V B	489	36513	35108	36219	**35141**	68050

Table A.5 (Cont'd) Average computational time (ms) requirements of circuit switching methods.

Network Type	Y	MG	MAG	MMG	MAMG	DLS
75 10 V UB	480	33261	31659	32669	**31685**	66095
75 25 0.3 B	1457	113255	115936	115061	**114153**	239527
75 25 0.3 UB	1417	105516	107115	101778	**106317**	237016
75 25 0.7 B	2835	143143	44740	103223	89273	**592640**
75 25 0.7 UB	2396	95648	40779	93428	40351	**515771**
75 25 V B	1540	139595	139727	134471	**140984**	235030
75 25 V UB	1485	125650	125232	120935	**126674**	227129
75 50 0.3 B	3916	264982	251397	244298	**268375**	498899
75 50 0.3 UB	3587	253226	243440	231164	**279711**	497148
75 50 0.7 B	7380	252984	123953	268417	127728	**1677364**
75 50 0.7 UB	5936	203112	106098	198150	101409	**1180746**
75 50 V B	3716	265629	272986	258051	**234845**	426876
75 50 V UB	2863	210476	210267	202572	**213977**	366612
75 100 0.3	6349	370168	383723	400696	**378015**	858405
75 100 0.7	15888	338978	195815	338484	194012	**2396452**
75 100 V	3547	465470	418773	450262	**348789**	716995

Table A.6 Quality comparison of packet switching methods

AMG: Adapted Minoux Greedy **ADLS**: Adapted Disaggregate Local Search

Network Type	CBE	Gersht	CLE	CLE Inc	AMG	ADLS
25 10 0.3 B	32,41 ± 1,96	**0,23 ± 0,19**	2,53 ± 0,50	3,42 ± 0,60	0,57 ± 0,24	6,99 ± 1,13
25 10 0.3 UB	30,02 ± 1,72	**1,25 ± 0,62**	3,25 ± 0,77	3,98 ± 0,73	1,74 ± 0,63	4,02 ± 1,15
25 10 0.7 B	5,18 ± 0,74	4,18 ± 0,82	1,70 ± 0,42	1,67 ± 0,42	5,48 ± 0,63	**0,39 ± 0,20**
25 10 0.7 UB	4,83 ± 0,66	2,67 ± 0,63	1,50 ± 0,38	1,52 ± 0,38	5,70 ± 0,67	**0,65 ± 0,29**
25 10 V B	76,15 ± 9,58	81,94 ± 12,30	94,34 ± 13,80	85,11 ± 12,80	80,37 ± 12,43	**0,03 ± 0,12**
25 10 V UB	86,01 ± 10,02	93,63 ± 13,33	103,1 ± 14,12	97,68 ± 14,01	93,30 ± 13,55	**0,01 ± 0,04**
25 25 0.3 B	37,06 ± 2,28	**0,19 ± 0,10**	4,58 ± 0,60	4,72 ± 0,66	0,84 ± 0,24	20,44 ± 1,30
25 25 0.3 UB	36,74 ± 2,09	**0,17 ± 0,09**	4,60 ± 0,63	4,88 ± 0,59	0,73 ± 0,21	19,93 ± 1,33
25 25 0.7 B	3,69 ± 0,51	2,37 ± 0,53	0,67 ± 0,24	**0,52 ± 0,19**	6,56 ± 0,57	2,07 ± 0,36
25 25 0.7 UB	3,54 ± 0,51	1,75 ± 0,48	0,83 ± 0,26	**0,64 ± 0,23**	6,94 ± 0,56	2,01 ± 0,35
25 25 V B	19,25 ± 4,61	19,28 ± 5,4	30,77 ± 6,19	16,72 ± 5,19	9,82 ± 4,16	**2,54 ± 0,99**
25 25 V UB	20,43 ± 4,38	22,45 ± 5,64	32,70 ± 6,33	21,12 ± 5,32	13,27 ± 4,72	**2,10 ± 0,88**
25 50 0.3 B	28,92 ± 2,23	**0,18 ± 0,12**	3,34 ± 0,53	3,40 ± 0,50	1,04 ± 0,26	19,52 ± 1,16
25 50 0.3 UB	28,28 ± 2,09	**0,27 ± 0,13**	3,67 ± 0,60	3,78 ± 0,52	0,74 ± 0,23	19,26 ± 1,22
25 50 0.7 B	2,87 ± 0,36	2,35 ± 0,55	0,39 ± 0,13	**0,36 ± 0,15**	6,81 ± 0,51	2,42 ± 0,29
25 50 0.7 UB	2,58 ± 0,34	1,99 ± 0,48	0,46 ± 0,16	**0,34 ± 0,16**	6,72 ± 0,48	2,11 ± 0,28
25 50 V B	20,22 ± 4.03	17,51 ± 4,64	31,46 ± 5,86	15,18 ± 4,05	8,82 ± 3,52	**4,05 ± 1,54**
25 50 V UB	20,82 ± 3,96	18,07 ± 5,05	32,94 ± 6,07	16,73 ± 4,76	10,14 ± 3,98	**3,71 ± 1,49**
25 100 0.3	23,48 ± 1,96	**0,19 ± 0,11**	1,96 ± 0,38	1,88 ± 0,39	1,33 ±0,28	16,84 ± 1,05
25 100 0.7	2,47 ± 0,28	2,09 ± 0,49	0,31 ± 0,10	**0,14 ± 0,07**	6,42 ± 0,42	1,58 ± 0,19
25 100 V	32,51 ± 5,24	17,07 ± 5,42	37,34 ± 7,29	21,27 ± 5,58	16,72 ± 5,26	**0,24 ± 0,27**
50 10 0.3 B	54,01 ± 1,84	**0,03 ± 0,03**	6,29 ± 0,53	7,64 ± 0,57	0,95 ± 0,18	23,38 ± 0,85
50 10 0.3 UB	50,74 ± 1,83	**0,02 ± 0,02**	6,54 ± 0,53	7,56 ± 0,65	0,96 ± 0,20	21,58 ± 0,91
50 10 0.7 B	5,25 ± 0,44	4,43 ± 0,69	0,81 ± 0,23	0,69 ± 0,22	10,29 ± 0,51	**0,67 ± 0,18**

Table A.6 (Cont'd) Quality comparison of packet switching methods

Network Type	CBE	Gersht	CLE	CLE Inc	AMG	ADLS
50 10 0.7 UB	5,10 ± 0,46	3,57 ± 0,67	0,89 ± 0,26	0,82 ± 0,24	11,12 ± 0,57	**0,62 ± 0,19**
50 10 V B	16,13 ± 2,58	20,10 ± 3,07	29,15 ± 3,61	14,67 ± 2,51	6,87 ± 2,12	**0,60 ± 0,31**
50 10 V UB	21,76 ± 2,93	26,24 ± 3,30	35,92 ± 3,71	22,23 ± 3,07	13,71 ± 2,76	**0,19 ± 0,19**
50 25 0.3 B	33,90 ± 1,73	**0,10 ± 0,08**	4,59 ± 0,45	6,17 ± 0,53	1,27 ± 0,26	24,95 ± 0,95
50 25 0.3 UB	33,72 ± 1,61	**0,08 ± 0,07**	5,25 ± 0,46	6,55 ± 0,54	1,19 ± 0,26	25,34 ± 0,93
50 25 0.7 B	3,52 ± 0,35	2,93 ± 0,49	0,62 ± 0,19	**0,31 ± 0,14**	12,29 ± 0,49	2,36 ± 0,24
50 25 0.7 UB	3,42 ± 0,32	2,61 ± 0,52	0,50 ± 0,14	**0,33 ± 0,12**	12,79 ± 0,50	1,96 ± 0,25
50 25 V B	11,79 ± 1,46	13,82 ± 1,99	23,17 ± 2,73	7,92 ± 1,34	**0,68 ± 0,76**	3,71 ± 0,54
50 25 V UB	12,33 ± 1,49	13,10 ± 1,90	25,93 ± 2,60	8,83 ± 1,42	**1,02 ± 0,97**	3,63 ± 0,58
50 50 0.3 B	28,13 ± 1,33	**0,12 ± 0,08**	2,12 ± 0,32	3,21 ± 0,37	1,40 ± 0,27	22,96 ± 0,89
50 50 0.3 UB	27,83 ± 1,36	**0,12 ± 0,08**	2,39 ± 0,37	3,49 ± 0,40	1,24 ± 0,27	23,07 ± 0,95
50 50 0.7 B	3,00 ± 0,25	2,53 ± 0,48	0,53 ± 0,14	**0,17 ± 0,09**	12,18 ± 0,45	2,32 ± 0,21
50 50 0.7 UB	3,03 ± 0,26	2,46 ± 0,50	0,48 ± 0,13	**0,22 ± 0,10**	12,97 ± 0,46	2,15 ± 0,21
50 50 V B	13,08 ± 1,27	8,21 ± 1,72	20,77 ± 2,32	6,33 ± 0,79	**0,10 ± 0,18**	5,76 ± 0,61
50 50 V UB	13,43 ± 1,38	9,98 ± 1,68	21,74 ± 2,50	7,13 ± 0,96	**0,44 ± 0,35**	7,60 ± 1,35
50 100 0.3	23,94 ± 1,91	**0,28 ± 0,20**	0,93 ± 0,39	1,16 ± 0,41	1,73 ± 0,17	20,28 ± 1,34
50 100 0.7	2,40 ± 0,32	0,55 ± 0,19	0,46 ± 0,19	**0,13 ± 0,16**	11,78 ± 0,92	1,64 ± 0,29
50 100 V UB	17,35 ± 2,90	2,72 ± 1,53	21,13 ± 4,70	6,46 ± 1,78	2,38 ± 1,45	**2,14 ± 1,16**
75 10 0.3 B	45,60 ± 2,09	**0,02 ± 0,04**	7,17 ± 0,45	9,67 ± 0,58	1,33 ± 0,22	29,28 ± 0,81
75 10 0.3 UB	46,01 ± 2,20	**0,02 ± 0,03**	7,95 ± 0,51	10,00 ± 0,51	1,34 ± 0,20	28,42 ± 0,87
75 10 0.7 B	4,86 ± 0,38	4,57 ± 0,63	0,65 ± 0,19	**0,29 ± 0,13**	14,08 ± 0,43	1,96 ± 0,23
75 10 0.7 UB	4,51 ± 0,41	2,89 ± 0,52	0,64 ± 0,17	**0,45 ± 0,18**	14,94 ± 0,49	1,30 ± 0,25
75 10 V B	10,73 ± 0,91	16,93 ± 1,63	25,0 ± 1,91	8,26 ± 1,09	**0,81 ± 0,59**	3,47 ± 0,62
75 10 V UB	10,96 ± 1,01	17,07 ± 1,64	26,52 ± 2,15	9,91 ± 1,10	**1,06 ± 0,59**	2,44 ± 0,58
75 25 0.3 B	33,00 ± 1,1	**0,02 ± 0,04**	3,95 ± 0,35	6,13 ± 0,42	1,81 ± 0,27	28,02 ± 0,80

Table A.6 (Cont'd) Quality comparison of packet switching methods

Network Type	CBE	Gersht	CLE	CLE Inc	AMG	ADLS
75 25 0.3 UB	33,13 ± 1,09	**0,03 ± 0,03**	4,39 ± 0,36	6,81 ± 0,44	1,58 ± 0,26	27,89 ± 0,79
75 25 0.7 B	3,69 ± 1,34	3,49 ± 0,60	0,69 ± 0,15	**0,15 ± 0,08**	16,18 ± 0,49	2,44 ± 0,26
75 25 0.7 UB	3,75 ± 0,37	3,06 ± 0,54	0,69 ± 0,18	**0,25 ± 0,12**	17,57 ± 0,47	2,32 ± 0,25
75 25 V B	12,53 ± 0,82	13,81 ± 1,46	22,3 ± 1,87	7,34 ± 0,67	**0,00 ± 0,00**	7,46 ± 0,52
75 25 V UB	13,68 ± 1,07	13,30 ± 1,18	24,12 ± 1,72	8,66 ± 0,74	**0,10 ± 0,15**	7,14 ± 0,58
75 50 0.3 B	28,33 ± 2,26	**0,06 ± 0,10**	1,32 ± 0,47	2,77 ± 0,66	1,74 ± 0,53	24,78 ± 1,87
75 50 0.3 UB	28,92 ± 2,01	**0,09 ± 0,11**	1,57 ± 0,71	3,20 ± 0,83	1,59 ± 0,60	25,00 ± 1.67
75 50 0.7 B	2,80 ± 0,49	2,63 ± 1,10	0,54 ± 0,36	**0,17 ± 0,18**	16,62 ± 1,03	1,74 ± 0,45
75 50 0.7 UB	3,02 ± 0,57	2,93 ± 1,04	0,67 ± 0,40	**0,16 ± 0,16**	17,00 ± 0,90	1,81 ± 0,49
75 50 V B	13,99 ± 1,82	8,53 ± 3,72	20,41 ± 3,41	6,63 ± 1,77	**0,15 ± 0,52**	8,49 ± 1,25
75 50 V UB	14,89 ± 1,98	8,03 ± 2,76	19,22 ± 2,96	6,64 ± 1,93	**0,007 ± 0,02**	8,71 ± 1,23
75 100 0.3	24,06 ± 2,28	Not Tested	**0,02 ± 0,13**	Not Tested	1,94 ± 0,66	21,36 ± 1,93
75 100 0.7	1,94 ± 0,37	Not Tested	0,42 ± 0,30	Not Tested	15,13 ± 1,04	**0,27 ± 0,31**
75 100 V	15,77 ± 1,98	Not Tested	14.26 ± 2.82	Not Tested	**0.05 ± 0.14**	4.60 ± 1.43

Table A.7 Average computational time (ms) of packet switching methods

AMG: Adapted Minoux Greedy **ADLS**: Adapted Disaggregate Local Search

Network Type	CBE	Gersht	CLE	CLE Inc	AMG	ADLS
25 10 0.3 B	585	**430**	164	187	82	235
25 10 0.3 UB	563	**425**	147	175	74	268
25 10 0.7 B	389	342	110	127	63	**673**
25 10 0.7 UB	381	336	108	127	60	**200**
25 10 V B	359	446	135	218	84	**37**
25 10 V UB	361	402	114	205	83	**40**
25 25 0.3 B	1366	**3053**	1271	1804	404	2431
25 25 0.3 UB	1536	**2879**	1197	1702	378	2267
25 25 0.7 B	1203	2595	821	**1042**	362	2526
25 25 0.7 UB	1154	2491	831	**1097**	323	2507
25 25 V B	913	3156	1102	2485	436	**100**
25 25 V UB	895	2826	1004	2225	207	**100**
25 50 0.3 B	2565	**11037**	4863	7272	884	5409
25 50 0.3 UB	2453	**10753**	4533	7191	839	4997
25 50 0.7 B	2484	8359	3245	**4207**	800	6652
25 50 0.7 UB	2416	7942	3396	**4201**	750	6271
25 50 V B	1754	11067	4436	10734	862	**3959**
25 50 V UB	1720	10394	4240	9998	810	**3660**
25 100 0.3	5101	**38134**	17609	29540	1544	11817
25 100 0.7	5610	25707	12996	**16792**	1472	18236
25 100 V	3586	36892	37138	66049	1578	**9538**
50 10 0.3 B	8341	**32432**	10901	14111	4255	17464
50 10 0.3 UB	8756	**27084**	9843	13135	3858	16496
50 10 0.7 B	6213	21690	5492	6592	3441	**18411**
50 10 0.7 UB	6098	19680	5150	6498	3254	**18265**
50 10 V B	5023	30139	8258	24597	4404	**11786**
50 10 V UB	4507	27186	7295	20885	3834	**11647**
50 25 0.3 B	13582	**188101**	62659	93620	12563	43825
50 25 0.3 UB	13123	**162871**	42623	91229	10964	34741
50 25 0.7 B	13657	129312	41961	**48257**	11103	61948
50 25 0.7 UB	13553	121365	40539	**48688**	10446	58825
50 25 V B	10141	214447	60477	213794	**12879**	29956
50 25 V UB	10319	213787	54359	157991	**12039**	30723
50 50 0.3 B	27521	**685217**	317522	536774	29836	110345
50 50 0.3 UB	25248	**678291**	239534	492967	26654	98434
50 50 0.7 B	31165	607080	208710	**247842**	27473	207404
50 50 0.7 UB	30933	511785	205961	**248307**	35272	217325
50 50 V B	20611	717425	613770	1277154	**30294**	50546
50 50 V UB	20498	701614	287755	831765	**29842**	66131
50 100 0.3	37331	**2439425**	875837	1770192	24597	108184
50 100 0.7	49366	2420746	922611	**784993**	22747	219433
50 100 V UB	30785	2744897	923248	784993	22784	**220029**
75 10 0.3 B	12067	273578	67370	90937	19976	62240
75 10 0.3 UB	11806	249418	59017	84033	17622	59461
75 10 0.7 B	12440	192117	38035	**42577**	16232	81949
75 10 0.7 UB	12085	179811	33923	**39819**	14953	80561
75 10 V B	10173	256833	53932	178539	**19135**	40682
75 10 V UB	9166	239072	49361	173085	**17456**	39474

Table A.7 (Cont'd) Average computational time (ms) of packet switching methods

Network Type	CBE	Gersht	CLE	CLE Inc	AMG	ADLS
75 25 0.3 B	53993	**2086273**	650016	1052103	108927	205881
75 25 0.3 UB	36936	**1918897**	604201	1063565	100543	214329
75 25 0.7 B	65651	1727722	477616	**567420**	95952	488422
75 25 0.7 UB	64778	1627417	396781	**667440**	93350	566030
75 25 V B	53722	2798224	808162	3303387	**146348**	227373
75 25 V UB	42282	2058640	599214	2544078	**106459**	187743
75 50 0.3 B	100651	**9711897**	2743613	5499306	224292	436724
75 50 0.3 UB	98387	**8467110**	2762821	5199884	227373	380531
75 50 0.7 B	170920	10374845	2402039	**2676900**	270342	1683230
75 50 0.7 UB	135353	6846510	1520332	**1917042**	160729	1004648
75 50 V B	90053	9480570	2765113	12332508	**249383**	389638
75 50 V UB	85718	8761094	2228377	11088760	**198649**	499082
75 100 0.3	169663	No Test	**14073573**	No Test	198902	605440
75 100 0.7	238385	No Test	11211180	No Test	192449	**1556902**
75 100 V	268750	No Test	20575245	No Test	**497449**	976261