ROUTING OPTIMIZATION METHODS FOR COMMUNICATION NETWORKS

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

BY

AHMET EMRAH DEMİRCAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 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 the 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.

Prof. Dr. M. Kemal LEBLEBİCİOĞLU
Supervisor

Examining Committee Members

Prof. Dr. Uğur HALICI (METU EEE) _____

Prof. Dr. M. Kemal LEBLEBİCİOĞLU (METU EEE) _____

Asst. Prof. Dr. Cüneyt BAZLAMAÇCI (METU EEE) _____

Dr. Senan Ece GÜRAN (METU EEE) _____

Alper GERÇEK (ASELSAN) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Ahmet Emrah DEMİRCAN

Signature :

ABSTRACT

ROUTING OPTIMIZATION METHODS FOR COMMUNICATION NETWORKS

Demircan, Ahmet Emrah
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. M. Kemal Leblebicioğlu

January 2005, 104 pages

This study discusses the routing optimization techniques and algorithms for communication networks. Preventing data loss on overloaded communication links and utilizing link bandwidths efficiently are the main problems of traffic engineering. Load balancing and routing problems are solved using both by heuristics such as genetic algorithms, and simulation techniques. These algorithms work on destination-based or flow-based routing techniques and mainly change the link weight system or try to select the best routes set upon K-routes routing table respectively to optimize network utilization. In this study, first a definition of the network routing optimization problem will be made. Then the heuristics to solve the problem will be discussed and finally an analysis of these heuristics will be made on sample network models. This thesis includes a discussion about the performance of different optimization heuristics working as a part of the centralized network load balancing systems.

Keywords: Routing optimization, destination-based routing, flow-based routing, heuristics, genetic algorithms.

ÖZ

İLETİŞİM AĞLARI İÇİN YÖNLENDİRME İYİLEŞTİRİLMESİ YÖNTEMLERİ

Demircan, Ahmet Emrah
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. M. Kemal Leblebicioğlu

Ocak 2005, 104 sayfa

Bu çalışma, iletişim ağlarının daha etkin kullanılması için yönlendirme iyileştirilmesi yöntemlerini incelemektedir. İletişim hatları üzerinde oluşacak fazla yüklerin oluşturacağı veri kaybının azaltılması, varolan hatların kullanım oranlarının dengeli dağıtılması, yönlendirme yöntemlerinin başlıca sorunudur. İletişim ağları yük ayarlama ve yönlendirme sorunu, genetik veya benzeştirim algoritmaları gibi buluşsal yöntemler kullanılarak çözülmeye çalışılmaktadır. Bu algoritmalar, temel olarak, hedef bağımlı yöntemler için hat ağırlık tablolarının iyileştirilmesi, akış bağımlı yönlendirme yöntemleri için yönlendirme tablosundaki en iyi yolların seçilmesi işlemlerini gerçekleştirmektedir. Bu çalışmada, önce iletişim alarında yönlendirme probleminin tanımı yapılmış, sonra problemin çözümü için sunulan buluşsal yöntemler tanıtılmış ve en son başarım çözümlemesi sunulmuştur. İletişim ağları merkezi yönetim tekniği kullanılarak veri hatları üzerinde yük ayarlaması yöntemleri hedef veya akış bağımlı yönlendirme için ayrı ayrı tartışılıp başarım tahlilleri yapılmıştır. Yönlendirmenin iyileştirilmesi için kullanılan buluşsal yöntemlerin yapıları ve yetenekleri ile ilgili benzeştirim yazılımı ile yapılan denemeler ve sonuçları karşılaştırmalı olarak verilmiştir.

Anahtar Kelimeler: Yönlendirme iyileştirilmesi, hedef bağımlı yönlendirme, akış bağımlı yönlendirme, buluşsal yöntemler, genetik algoritmalar.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DLR	Demand loss rate
DRCSP	Dynamic routing control simulation program
EIGRP	Enhanced interior gateway routing protocol
GA(W)	Genetic algorithm weight system dependent
GA(K)	Genetic algorithm K-routes system dependent
ISP	Internet service provider
MPLS	Multi protocol label switching
OSPF	Open shortest path first
PGA(W)	Parallel genetic algorithm weight system dependent
QoS	Quality of service
SAN	Simulated annealing meta-heuristic
WA	Weights adjustment heuristic

CHAPTER 1

INTRODUCTION

Today's communication networks, providing multimedia services and having real-time communication capable infrastructures, are being examined thoroughly by traffic engineers because of increasing traffic demands of network users. Internet has been presenting an unprecedented evolution for the last decade. List of applications used on the Internet has been upgraded from a short specific application list to a very long and globally used application list. The number of Internet users has also increased from a relatively small number of specialists to a very large number of people on the world. Not only the applications evolved but also the number of users increased [9].

The technological improvements on the networking devices and users' needs have, in coordination, developed today's evolving Internet. Unfortunately, the speed of networking technology improvements slowed down considerably while applications' need for speed is still increasing. Thus, network service providers work on better utilization of resources because of cost-efficiency problems and economical difficulties of extending network resources. The need for using the existing network infrastructures as efficient as possible forces the providers to work on network planning and optimization densely.

New applications and services change the nature of traffic throughout the data networks having an impact on the amount of traffic and its distribution on the flow-level characteristics. Certainly, the percentage of real-time applications will continue increasing, where this increase will unfortunately result in new QoS constraints. The importance of network planning and optimization becomes clearer at this point. The consequences of network planning and optimization can be described in four titles [9]:

- Traffic quantity: In the last few years peer-to-peer file sharing, audio and video streaming application have lead to a huge amount of traffic increase on networks. This trend is expected to continue and it is observed that most of the ISP customers choose to hire high-speed network connections and make more peer-to-peer connections and audio and video streaming. Networks can be managed to handle capacity assignment and routing optimization in order to supply resources for the customer's demands as short-term solutions. It is also observed that the more the

demands served the more they increase. So, the only solution on the ISP side seems to be expanding the networks capacities regularly.

- Traffic distribution: It is expected the peer-to-peer applications make it difficult to predict the traffic distribution, due to the dynamic nature of applications. The fact that the traffic distribution cannot be predicted stresses the need for network optimization methods that are applied as network management tasks. Once the traffic distribution is observed, routing optimization can be applied to better utilize the network resources.
- Traffic flow characteristics: The asymmetric property of traffic needs to be considered during network planning and optimization processes. Especially for the dimensioning of networks this characteristic can be exploited for the sake of cost savings. By taking into account unidirectional links, the required capacities can be assigned more accurately. This distinction regarding direction is expected to have a greater impact in networks with lower traffic aggregates (such as in access networks) where the characteristics of individual flows have stronger influence on the overall traffic aggregate. In backbone networks the per-flow traffic asymmetry is less pronounced due to the superposition of a great number of flows.
- QoS constraints: It is obvious that QoS aspects, i.e., the various notions of QoS as well as the enabling technologies, have to be considered for network planning. They determine the constraints and the objective of the relevant optimization problems. As soon as QoS requirements have to be met, the networks need to be dimensioned appropriately. However, it is not possible to take into account every technological detail, as this would make the planning process too complex. Therefore, appropriate abstractions of the relevant technologies and QoS requirements are essential.

Dynamic and unexpectedly varying nature of current Internet traffic demands requires that the communication networks must be planned and managed for better utilization performance and service quality. Application of QoS constraints on communication networks is the main motivation of traffic engineering for design and proposal of congestion free networking techniques. Extension of network capabilities is always required as the network user demands are increasing with the new communication applications. But, building such an extended network requires a lot of money and extended network would not even be utilized well except the rush hours. Investment on a super-network may be loss of money for the communication service providers. Thus, optimization of current network resources and

delaying the network extension projects to a later time may seem logical until current network resources cannot provide QoS in spite of the routing optimizations made.

The set of proposed techniques for online network optimization can be called as dynamic routing control (similar name is also mentioned in [1]). Consequently, dynamic routing control serves for QoS based routing and, in this study, it will be clarified that it is nothing but a network state dependent adaptive routing control system that is used to optimize operational networks.

Dynamic routing control is carried out to hopefully resolve or reduce potential network performance problems in cases where increasing traffic loads or temporary traffic variations cause problems such as link congestions. It can either directly modify the routing tables, or indirectly change administrative link weights of communication links for changing the routing tables by the help of routing protocols. Dynamic routing control is carried out to minimize the network cost with the network resources in hand.

Dynamic routing control's performance has a strong dependency on various optimization and recovery methods proposed in its context. The time spent for optimization and recovery processes causes data loss and communication delays in transmission. This contradicts with QoS based routing. The service degradation faced with, results in customer dissatisfaction and profit loss. Thus, the optimization and recovery routines used should recover the network from a costly state within a short response time. Network routing optimization, which is dynamically applied online to the network, on a short-term (some minutes) or mid-term (some hours) time basis is the main motivation of this study.

In this study, the structure of dynamical routing control and its usage in Internet traffic engineering will be discussed first. QoS based routing and network state dependent adaptive routing concepts will be mentioned while explaining the relationship between these concepts and the traffic engineering styles. In the third chapter, network routing optimization problem will be defined and a discussion of the mathematical problem will be made. In the fourth chapter, optimization heuristics for solving the routing optimization problem will be explained by making an analogy between dynamical routing control and the optimization methods. The experimental work and performance analysis of optimization techniques on sample networks will be made in the fifth chapter. Every step for developing a dynamical

routing control system will be stated in order from motivation to implementation and analysis phases throughout the study.

CHAPTER 2

DYNAMIC ROUTING CONTROL

2.1 Dynamic Routing Control

Internet traffic engineering is defined as that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational networks. Traffic engineering encompasses the application of technology and scientific principles to the measurement, characterization, modeling, and control of Internet traffic [2, 3]. The major objective of Internet traffic engineering is optimizing the performance of an operational network, at both the traffic and resource levels. This is accomplished by covering the traffic performance requirements of network users, while utilizing network resources economically and reliably. Delays, delay variations, packet loss ratios and throughput measurements are all in the context of traffic performance requirements.

The most significant function of Internet traffic engineering is the routing of traffic from ingress nodes to egress nodes. This includes the control and the optimization of the routing tables, steering the demands through the network in the most effective way. The most important performance indicator of the networks is the feedback of the end users. This point must always be considered while developing traffic engineering policies and optimization mechanisms. The characteristics visible to the end users are the emergent properties of the network when viewed as a whole. Special care must be taken while choosing the network performance measures to optimize. Optimizing the wrong measures may achieve certain local objectives, but may fail and have disastrous consequences on the emergent properties of the network and thereby on the QoS perceived by end users of network services [4].

QoS-based routing is being recognized as the missing piece in the evolution of QoS-based service offerings in the Internet [5]. Internet traffic engineering has mostly been dealing with the QoS based routing and the related methods to optimize utilization of operational networks. QoS-based routing is defined as the routing method, which determines the paths for flows based on the network resource availability knowledge and as well as the QoS measures of each flow. The set of service requirements to be met by the network while transporting a flow are strictly defined by the flow's QoS measures, and the network's current resources that are controlled by the QoS-based routing methods must cover all these

measures. QoS-based routing can also be called as the set of objectives for dynamic routing control. The main objectives of the QoS based routing are as follows [5]:

- Dynamic determination of feasible paths: QoS based routing can dynamically determine a path, among many possible alternatives, that can accommodate the QoS requirements of the flow. Feasible path selection may be subject to the network optimization problem constraints.
- Optimization of resource usage: A network state dependent routing method can reduce the service degradations and aid in the efficient utilization of the network resources while improving the total network throughput.

- Type-of-service specific routing: In multi-service networks two different notions of routing optimization exist, depending on the capabilities of the employed routing technology [9]:
    - Service specific routing: *"Packets belonging to different traffic flows with equal source and destination addresses might traverse the network along different paths. The routes could be selected by taking into account the type of traffic and its QoS requirements (QoS routing). From a routing optimization point of view, service-specific routing in multi-service networks is similar to regular routing in networks with only one type of service. Since the routers treat the individual services independently, their routing patterns could also be optimized independently. Thus, routing optimization strategies for single services could be applied to each service in a multi-service environment."* [9]
    - Service unaware routing: "*If routing is not service specific, rerouting always affects all traffic classes equally. Therefore, routing optimization has to consider the effects of route changes on each of the traffic classes. It could be possible that QoS is improved for one service while it greatly deteriorates for the other since the bandwidth allocation scheme is not appropriate anymore after routing optimization."* [9]:
- Graceful performance degradation: State-dependent routing methods can compensate for transient inadequacies in network engineering especially during focused overload conditions, giving better throughput and a more graceful performance degradation as compared to a state-insensitive routing methods.

Network state dependent adaptive routing is another concept, which has very similar goals as those of QoS-based routing. In fact, adaptive routing has long been applied on circuit switched networks and now it has application areas on ATM networks, which has different traffic characteristics and topology from those of circuit-switched networks. Thus adaptive routing methodology must be studied with care and problems encountered and the related solutions must be learned. Network state dependent adaptive routing methods fundamentally work in two steps: "collecting network state information" and "computing the feasible paths based on the given network state information" [5].

QoS-based routing and network state dependent adaptive routing are two fundamental blocks building the structure of dynamic routing control. Those two fundamental blocks and their relation with the routing optimization methods will be discussed throughout the paper. As a first step, some styles and strategies used for adaptive QoS-based routing methods in traffic engineering will be dealt in the next section.

2.1.1 Routing Control Related Internet Traffic Engineering Concepts

Internet traffic engineering systems are built on some mechanisms and methodologies that effect different aspects of a dynamic routing control implementation and its performance on the system. Some of these concepts determine the fundamental implementation styles, where the others define the position of dynamic routing control in Internet traffic engineering methodologies. Dynamic routing control will be directly related to the following mechanism and methodology classes of Internet traffic engineering:

- Time-dependent vs. State-dependent Network Control Mechanisms
- Offline vs. Online Routing Control Mechanisms
- Centralized vs. Distributed Network Management Mechanisms
- Destination-Based vs. Flow-Based Routing Strategies
- Local vs. Global Information Sampling Mechanisms

Below, these concepts are first explained and then dynamic routing control's relationship with each concept is told.

2.1.2.1 Time-dependent vs. State-dependent Control Mechanisms

In traffic engineering, network control methodologies can be classified as time-dependent and state-dependent. In the time-dependent control methods, historical information based on periodic variations in traffic, (such as time of day), is used to pre-program routing plans and other control mechanisms. Additionally, customer subscription or traffic projection may be used. Pre-programmed routing plans typically change on a relatively long time scale. Time-dependent algorithms do not attempt to adapt to random variations in traffic or changing network conditions.

On the other hand, state-dependent control methods adapt the routing plans for packets based on the current state of the network. The current state of the network provides additional information on variations in actual traffic (i.e., perturbations from regular variations) that could not be predicted using historical information. An example of a state-dependent control mechanism is a global centralized optimizer where the input to the system is a traffic matrix and multi-class QoS requirements as described [6].

Time-dependent methods rely on the observed network traffic values and make the assumption that the real-time values do not deviate from the observed values very much. All sampled time periods are examined in detail and least cost routing tables are built for each time period. In run-time, routing tables are always updated so that least-cost network is obtained.

State-dependent methods make no prediction of traffic values of the network. Each time instance of the network is observed, searching for an increase in the network cost. If a bad-utilized network is observed, the optimization problem is solved right at that moment and routing tables or routing protocol parameters such as administrative link weights are updated according to the results. State-dependent mechanisms can either be adapted to work with destination-based or flow based routing systems, where administrative link weights for shortest path determination process of routing protocols, or directly the routing tables of the routers are to be updated respectively.

Both mechanisms have advantages and disadvantages coming from their nature. These advantages and disadvantages can be classified according to the following two views:

- Adaptation to unexpected traffic variations: Time-dependent methods change the routing table according to the traffic patterns expected at each scheduled time, and

therefore cannot accommodate the unexpected traffic variations appearing on the network. State-dependent methods are flexibly adaptive to dynamic traffic variations because they generate new routing tables in real-time according to the changes in the monitored traffic. State-dependent methods are better than time-dependent methods considering adaptation to unexpected traffic demands. [1]

- Processing difficulty levels: Though the adaptive success of state-dependent methods, these methods accommodate a problem that they require a huge amount of processing time for implementation. State-dependent methods require online optimization of the routing tables and recalculation of routing tables for each traffic variation monitored on the network, where time-dependent methods use offline-optimized routing tables for each scheduled time, which make them faster. On the other hand, real-time traffic measurement and collection requirement of state-dependent control mechanisms is another difficulty on the usage of state-dependent control mechanism. Probing the network and collecting information from nodes requires extra bandwidth, which can, in fact, may result in network congestion in some networks. Usage of virtual control channels and giving priority to this control information in the network is a must. However, probing methods, probing refresh rate and the amount of probe information exchanged must be adjusted with care.

Using state-dependent methods is needed in today's communication networks but serious attention is required for making state-dependent methods faster. Although time-dependent methods work well under estimated traffic loads, there are times that estimation may fail in unexpected traffic load occurs. Thus, tendency in dynamic routing control method designs is towards finding a hybrid method having both adaptive character of state-dependent methods and fast response time of time-dependent methods.

2.1.2.2 Offline vs. Online Routing Control Mechanisms

Traffic engineering requires the computation of routing plans. The computation may be performed offline or online. The computation can be done offline for scenarios where routing plans need not be executed in real-time. For example, routing plans computed from forecast information may be computed offline. Typically, offline computation is also used to perform extensive searches on multi-dimensional solution spaces. Online computation is required when the routing plans must adapt to changing network conditions as in state-dependent mechanisms. Unlike offline computation (which can be computationally demanding and

time-consuming), online computation is geared toward relative simple and fast calculations to select routes, fine-tune the allocations of resources, and perform load balancing. [4]

As it's known from the previous section, tendency is towards hybrid time-state-dependent methods where adaptation and fast solution convergence (response time) parameters are optimized. As it can easily be seen that time-dependent methods make use of offline routing where state-dependent methods make use of online routing.

2.1.2.3 Centralized vs. Distributed Network Management Mechanisms

Network management systems are classified into two groups as centralized and distributed. Centralized control has a central authority, which determines routing plans and perhaps other traffic engineering control parameters on behalf of each network node. The central authority collects the network-state information from all routers periodically and returns the routing information to the routers. The routing update cycle is a critical parameter directly impacting the performance of the network being controlled. Centralized control may need high processing power and constant bandwidth control channels. Distributed control determines route selection by each router autonomously based on the routers view of the state of the network. The network state information may be obtained by the router using a probing method or distributed by other routers on a periodic basis using link state advertisements. Network state information may also be disseminated under exceptional conditions. [4]

Dynamic routing control should use centralized network management mechanism to probe and maintain all network parameters such as routing tables. It has the advantage of maintaining whole network's QoS-based routing despite its processing and control bandwidth requirements. But, at network design time, control channels with convenient bandwidth and delay properties may be reserved. It is clear that distributed network management mechanism may fail in covering the end-to-end user QoS requirements because of its distributed and network-state-unaware routing methodology.

2.1.2.4 Destination-Based vs. Flow-Based Routing Strategies

Another traffic engineering related concept, which can be discussed in this section, is the routing strategy to be used in the network. There are two routing strategies used in traffic engineering [8, 9]:

- Destination-based routing
- Flow-based routing

Dynamic routing control adjusts the network routing tables so that available network resources are better utilized and a certain QoS is acquired. When it comes to management of routing tables, various concepts or methods can be proposed to carry out the routing optimization problems. Considering mentioned routing methods, it must be decided which one of these methods is appropriate for routing optimization problem. Discussing on the main principles of these two methods can help making this decision.

Destination-based routing methods, ones that are using OSPF or EIGRP follow the next hop determination procedure that is very fast but also limited on achieving good results from routing optimization procedures. Those destination-based routing algorithms can easily cause traffic overloads on some links and leave others lightly utilized. The main reason underlying this utilization problem is that every packet for the same destination (no matter what their source is) entering a router is routed through the same network interfaces that current routing protocol forces. As a solution to this problem, flow-based routing methods have been developed. Destination-based routing methods rely on the administrative link weights given for each link in the communication network. Routing optimization processes for destination-based routing enabled networks modify these administrative link weights in the network. Link weight adjustment process in the optimizer results in a hopefully least cost network state, and the network management system updates all weight tables to let the routing protocol do the messaging and reconstruct the shortest paths at each router. To sum up, in destination-based routing methods dynamic routing control never plays with the routing tables directly and routing tables are updated by the routing protocols

Flow-based routing strategy, depending on connection-oriented routing (using MPLS capabilities or ATM technology) setup explicit paths for each flow of different source-destination pairs so that a high degree of routing freedom is obtained. There are many network configurations that can be achieved by changing the selected path of one source-destination pair while fixing the remaining source-destination pairs. This freedom in flow-

based routing secures fair distribution of demands on links, thus a better-utilized network is obtained. Routing optimization processes for flow-based routing enabled networks directly modify the routing tables at each router in the network.

It seems better to use flow-based routing because of this concept's contribution to the speed of optimization heuristics and utilization friendly characteristics. Various optimization techniques based on flow-based routing concept are now being adapted to online routing optimization problems in order to achieve desired QoS constraints through real-time management of communication networks. However, there are still enough convincing points to use destination-based routing concept in network management. In [22], the answer to the question *"Can a sufficiently clever weight settings make OSPF routing perform nearly as well as optimal general/MPLS routing?"* is given as "No., but…". Insisting on the use of destination-based routing concept is explained as follows:

"The next natural question is: how well does OSPF routing perform on real networks. In particular we wanted to answer this question for a proposed AT&T WorldNet backbone. In addition, we studied synthetic inter-networks, generated as suggested by Calvert, Bhattacharjee, Daor, and Zegura [5], [6]. Finding a perfect answer is hard in the sense that it is NP-hard to find an optimal setting of the OSPF weights for an arbitrary network. Instead we resorted to a local search heuristic, not guaranteed to find optimal solutions. Very surprisingly, it turned out that for the proposed AT&T WorldNet backbone, the heuristic found weight settings making OSPF routing performing within a few percent from the optimal general routing. Thus for the proposed AT&T WorldNet backbone with our projected demands, and with our concrete objective function, there would be no substantial traffic engineering gain in switching from the existing well-tested and understood robust OSPF technology to the new MPLS alternative." [22]

Switching from destination-based routing methods such as OSPF or EIGRP to flow-based routing technologies such as ATM (any connection oriented technology) is a popular discussion topic nowadays. Despite of the advantages of flow-based routing over destination-based routing, traffic engineers think twice before going for a system change and make experiments on the current communication networks to see the results.

2.1.2.5 Local vs. Global Information Probing Mechanisms

Traffic engineering algorithms may require local or global network-state information. Local information pertains to the state of a portion of the domain. Examples include the bandwidth and packet loss rate of a particular path. Local state information may be sufficient for certain instances of distributed-control systems. Global information pertains to the state of the entire domain undergoing traffic engineering. Examples include a global traffic matrix and loading information on each link throughout the domain of interest. Global state information is typically required with centralized control. Distributed-control systems may also need global information in some cases. [4]

Dynamic routing control should use a global information probing method because of its centralized-management mechanism. State dependent, centralized management mechanisms would easily fail in the lack of network state information that is processed in a centralized-management node.

2.2 Structure of Dynamic Routing Control

Routing control is a significant aspect of Internet traffic engineering. Routing impacts many of the key performance measures associated with networks, such as throughput, delay, and utilization. Generally, it is very difficult to provide good service quality in a wide area network without effective routing control. A desirable routing system is one that takes traffic characteristics and network constraints into account during route selection while maintaining stability.

"In order to achieve higher QoS, based on a given network infrastructure and a certain traffic load situation, a network service provider might have the possibility to better distribute the traffic in the network. By reducing the link load values in the network, overload situations can be avoided. This can be done relying on conventional routing protocols such as OSPF and EIGRP, or by deploying new technologies such as MPLS.

The idea of routing optimization is to find a routing pattern, which achieves the best network QoS for a certain demand matrix (predicted or measured). Based on monitoring events or traffic forecasts, routing is changed from time to time. However, every time routing is reconfigured, affected traffic flows might experience service degradations due to short-term

transition phases and routing instabilities. Therefore, if routing is adapted frequently, the impact of rerouting has to be kept as low as possible. The number of rerouted flows or the amount of rerouted traffic needs to be taken into account when computing a new routing pattern. On the other hand, if routing optimization is done only on a very coarse time granularity basis, QoS optimality is the main criterion." [9]

Within the previous sections, routing related traffic-engineering concepts were discussed and an analogy between these and dynamic routing control was made. Figure 1, [10], will help visualizing the structure of a dynamic routing control.

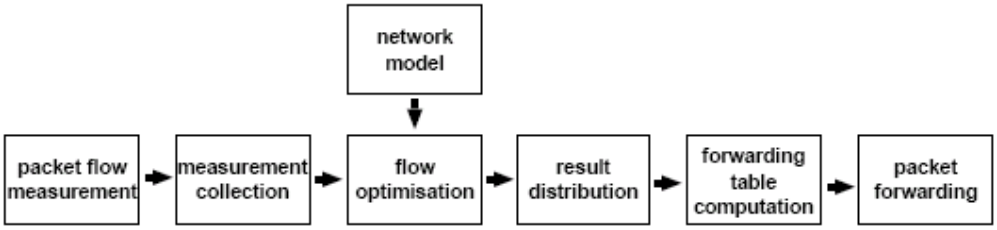


Figure 1: Structure of dynamic routing control

Let us explain how this structure fits into complete routing control architecture:

- Flow measurements at all ingress nodes are collected in the global demands matrix. QoS and BW requirements of the flows are collected in different matrices. Then these matrices are sent to the optimizer at regular time intervals. Probing the network in such a way makes the centralized flow optimizer aware of an estimate of the current network state. Those measurements are collected also to statistically make an estimation of traffic changes throughout the day. Estimated traffic values can be used in time-dependent routing with offline routing optimization methods. Flow measurement strategy should be chosen properly in order to optimize the performance of the dynamic routing control mechanism. Special care must be taken in refresh rate selection, traffic aggregation strategy (type-of-service specific routing) and optimization of the size of the collected data over the network. In [9], flow measurement strategy is explained as follows:

  "The measurements provide an estimate of the current demands matrix to the centralised flow optimisation. The demand matrix is aggregated at the level of all

traffic from an ingress node destined for a certain egress node. If a more fine-grained control over the traffic flows are desired, for instance to provide differentiated quality of service, a more fine-grained aggregation level can be chosen. This results in more commodities in the optimisation, which can be potential performance problem. One approach is to introduce two levels in the optimisation, one with a longer time-scale for quality of service flows. The flow measurement need to be averaged over a long enough time to get sufficiently stable values. Our current research as well as others [5] indicate that the needed stability exists in real networks at the timescale of a few, maybe five to ten, minutes." **[9]**

- Optimizer uses the network model, together with the global network information to build the optimum routing or administrative link weight tables, which utilize the network best. Optimization can be done according to type-of-service specific routing as mentioned in section 2.1. There are two routing optimization alternatives of the optimizer according to global network model:
    - Using offline optimization (time-dependent nature)
    - Using online optimization (state-dependent nature)

  Optimizer can use offline optimized routing tables or try to decrease network cost by online optimization if deviation from estimated traffic value does exist. Thus, optimizer can behave as both time-dependent and state-dependent according to the traffic measurements. The aim of the optimizer is to minimize the changes in routing tables as they result in short-time service degradations.

- Routing or administrative link weight tables at all nodes are updated.

As mentioned before, dynamic routing control used in network routing optimization problems must be hybrid so that it is network state-dependent and has fast solution convergence and response time. Given a network topology, under certain QoS measures an optimum routing solution must be served as soon as possible, because every time tick in a congested network means loss of data and bad utilization among the network. Thus, dynamic routing control should choose the quasi-optimal solution from the solution space in a small response time. Solution space for destination-based routing methods is the set of possible administrative link weight tables and solution space for flow-based routing methods is the set of possible routing tables realizing the demands. This requires the proposed method to be

effective and practical where it is decided that a hybrid management method consisting of time-dependent and state-dependent control characteristics is needed. State-dependent control feature of dynamic routing control is provided by regularly refreshed global network information and online optimization, and time-dependent control feature is provided by historical measurements, current global network information and offline optimization. Dynamic routing control can choose to use offline routing tables if current demands matrix and the average demands matrix do not differ from each other. Otherwise, it should make online optimization to hopefully decrease the network cost.

Considering the routing strategies, it has been seen that flow-based routing method, where the freedom of fair demand assignments and better utilization performance are available, is recommended. Flow-based routing methods relying on connection oriented routing also uses the advantage of differentiating type-of-service specific demands between same source and destination nodes with different aggregation levels and using different paths for them. However, destination-based routing algorithms cannot handle such a fine-grained aggregation level. On the other hand, robust structure of routing protocols such as OSPF and EIGRP and satisfactory results of destination-based routing on some networks convince traffic engineers to use this destination-based routing in network management (See the quotation in section "2.1.2.4 Destination-Based vs. Flow-Based Routing Strategies").

Considering the routing optimization problem of a huge network, the requirements for a control mechanism are critical. Let's now, summarize the fundamental properties of dynamical routing control:

- It should be aware of global network information,
- It should be state-dependent but should be fast as if it is time-dependent (fast response time),
- It should make both offline and online-routing,
- It should better use centralized-management mechanism,
- It should adapt to both destination-based and flow-based routing on different networks.

There have been many methods proposed for the solution of network optimization problems. These methods differ from each other if optimization and network management techniques used in each of them are examined. The methods using the same management techniques may also differ from each other because of the optimization technique differences. No matter

which management techniques used, another important factor on the method performance is the optimization technique efficiency and speed. There are many optimization techniques proposed for the network routing optimization problem, most of which are genetic algorithms and some other heuristics. Since the optimization technique used is an independent factor besides the management and routing techniques, network optimization problem will be discussed first, and optimization techniques will be discussed next.

CHAPTER 3

NETWORK ROUTING PROBLEM

3.1 Introduction

This and the next chapter include both theoretical (mathematical) and practical approaches on the network routing problem. Mainly, routing optimization plays a crucial role for network planning (initial design and extension planning) as well as for network operation (traffic engineering). Given a certain traffic demand matrix with QoS constraints (i.e. delay constraint and bandwidth requirements), it is the objective to minimize network cost (i.e. packet loss rate, max link utilization) only by modifying the routes of traffic flows through the network and not by changing or extending the network infrastructure. This optimization problem is called the network routing problem. Before going on examining solution methods of the network routing problem, the problem must be studied in depth. In the next two sections, definition of network is made firstly, and the problem is to be explained secondly. All abbreviations made in this chapter will also be used in the next chapters.

3.2 Definition of a Network and its Mathematical Representation

In this section, first the definition of network is made, some specific examples are given and then a mathematical approach to the definition is made. Since this work specializes in optimization of communication networks, the definition of communication networks will be made. This section includes some words like graph, directed or connected graph, vertex and edge, which are assumed to be known beforehand. Thus, the definitions, if included, are kept simple.

Definition of network is usually made as “a wide variety of systems of interconnected components”. In mathematics, a network is usually defined as a graph. Actually, network theory is the applied mathematics counterpart of the graph theory. General-purpose mathematical models of network structures and associated algorithms have been developed in graph theory. Computer network routing is a direct application of graph theory in the real world.

A network is modeled as a directed and connected graph $G(N, E)$ where N is a finite set of vertices (network nodes) and E is the set of edges (network links) connecting these vertices. The definitions of graph, connected and directed graph are made as follows:

- Graph is a generalization of the simple concept of a set of dots, called vertices (or nodes), connected by edges (or links).
- Connected Graph is a graph such that there is a path between any pair of nodes via zero or more other nodes. Thus starting from any node and visiting all nodes connected to that node by a single edge, then going for all nodes connected to any of the edges, and so on, eventually every node in the connected graph can be visited.
- Directed Graph (digraph) is a graph with one-way edges (the edges have a direction associated with them)

Let's continue the discussion with elements corresponding to graph $G(N, E)$. Let:

- $n = |N|$ be the number of vertices of network $G(N, E)$
- $l = |E|$ be the number of edges of network $G(N, E)$
- $e(u, v)$ be the link from node u to v which implies the existence of link $e'(v, u)$ from node v to u.

In addition to the above parameters, there are five functions associated with each link $e (e \in E)$ of the network:

- $C(e): E \to R^+$: The link cost function, $C(e)$, may be either a monetary cost or any measure of resource utilization, which must be optimized.
- $D(e): E \to R^+$: The link delay function, $D(e)$, is the sum of switching, queuing, transmission and propagation delays that occur between two connected nodes $u, v \in N$.
- $B(e): E \to R^+$: The link bandwidth, $B(e)$, is the bandwidth of the physical or logical link $e(u, v) \in E$ connecting nodes $u, v \in N$
- $UB(e): E \to R^+$: The used link bandwidth, $UB(e)$, is the used bandwidth of the physical or logical link $e(u, v) \in E$ connecting nodes $u, v \in N$

- $RB(e):\ E \rightarrow R^{+}$: The residual link bandwidth, $RB(e)$, is the residual bandwidth of the physical or logical link $e(u,v) \in E$ connecting nodes $u, v \in N$ at any instant of the communication. $RB(e)$ can take a value in between $B(e)$ and 0:

$$RB(e) = B(e) - UB(e);\ B(e) > UB(e) : E \rightarrow R^{+}$$

The link delay $D(e)$ and bandwidth $B(e)$ functions define the criteria that must be constrained. Because of the asymmetric nature of the communication networks, it is often the case that:

$$C(e) \neq C(e') \qquad e(u,v) \in E\ ;\ e'(v,u) \in E\ ;\ u, v \in N$$
$$D(e) \neq D(e') \qquad e(u,v) \in E\ ;\ e'(v,u) \in E\ ;\ u, v \in N$$
$$B(e) \neq B(e') \qquad e(u,v) \in E\ ;\ e'(v,u) \in E\ ;\ u, v \in N$$

As a result, a network consists of nodes and links (vertices and edges) where each node can source traffic to other nodes and this traffic is routed through the links with subject to some constraints. In the next section, the problem related to traffic routing in networks is discussed.

3.3 Definition of the Network Routing Optimization Problem

Communication networks contain nodes sourcing demands destined for other nodes, where each demand requires QoS constraints (delay, bandwidth constraints and related objective functions). Every demand is routed via zero or more nodes subject to the demands' delay and bandwidth constraints. Each routed demand consumes the network resources and conforms to the current network resource availability. There is always a chance that the demands may not be covered totally because of the current network resources and routing tables. This is called service degradation. But a network that is appropriately planned for possible traffic demands still has the chance to recover from service degradation. This is accomplished by managing the routing tables or route reservations while adapting to the dynamic traffic demands. At each time instance of the communication throughout the network, total cost of the network is evaluated and an optimization process is run to minimize this cost. In the general routing problem, there are no limitations on how flows can be distributed along the paths from source to destination, and the problem can be formulated and solved in

polynomial time as a multi commodity flow problem. This whole process is carried out to solve the network routing optimization problem. In the following paragraphs, some network routing problem classes are explained.

Routing problems can be classified into two main classes as unicast and multicast routing. In this study, unicast routing is handled. The unicast routing problem is defined as follows: given a source node s and a destination node t, a set of QoS constraints C, and possibly an optimization goal, find the best feasible path from s to t which satisfies C. This problem can also be classified into subclasses according to the QoS constraint set C as follows: [23]

- Link QoS: For some QoS metrics such as residual bandwidth or residual buffer space, the state of a path is determined by the state of the bottleneck link. Thus optimal routing is always link QoS related. There are two subclasses of this problem:
    - Link-optimization routing: Link optimization routing problem is defined as finding a path that has the largest bandwidth on the bottleneck link (widest path).
    - Link-constrained routing: Link constrained routing is on the other hand defined as finding a path whose bottleneck value is above a required value.
- Path QoS: For some QoS metrics such as delay, delay jitter, and cost, the state of a path is determined by the combined state over all links on the path. Two basic routing problems can be defined for this kind of QoS metrics:
    - Path-optimization routing: Path-optimization routing is defined as finding a path whose total cost is minimized.
    - Path-constrained routing: Path constrained routing is on the other hand defined as finding the path whose total QoS constraint (i.e. delay) is bounded by a required value.

Many composite routing problems can be derived from the above four problems. Some of these problems are solvable in polynomial time and some of them are NP-complete. Figure 2 shows these composite problem classes. In this study, path-constrained path-optimization composite routing problem is studied and this will be described in the next section.

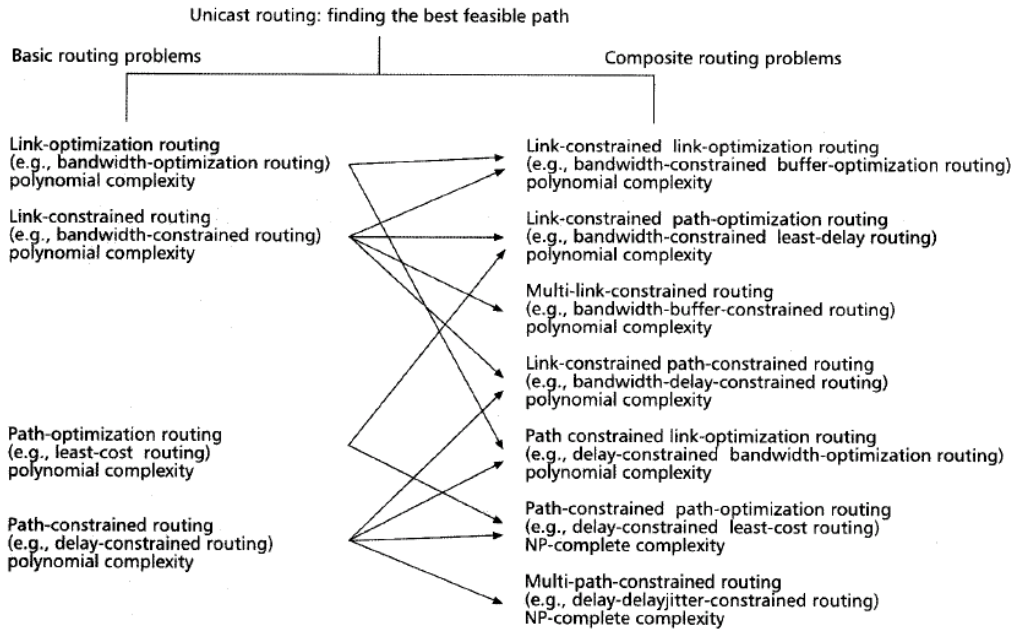


Figure 2: Unicast routing problem classes, [23].

The objective function of network routing problem is strictly dependent on the QoS requirements of traffic demands. QoS requirements are defined as the network service requirements that must be covered in transmission. Each traffic flow that is transported in the network forces the network resources to cover its own QoS measure. This is called *flow-based QoS*. But, if QoS requirements shall be used as an objective function in network, managing all traffic flow measures is faced with and presenting a new, overall measure of QoS that fits on a network configuration is required. This new measure covering all flows requirements is called the *network QoS*.

Let's now inspect these two types:

- Flow-based QoS: This can be measured by using packet delays, packet loss rates and throughput values:
    - Packet Delay: End to end delay of a packet in the transportation from source to destination is called the packet delay. Packet delay is the sum of all processing, queuing and propagation delays encountered on the path. Real-time traffic is prone to packet delays while non-real-time data traffic is not. Real time applications require short and constant delays in transportation.
    - Packet Loss Rate: Both real-time services and non-real-time data applications are prone to packet loss rate. Increasing packet loss rate in non-

real-time data traffic causes retransmission of packets, which slows the overall transfer process and decreases the QoS. On the other side, packet losses in real-time traffics cannot be recovered by retransmissions. Thus increasing packet loss rate causes unacceptable service degradation.

  - Throughput: This measure is often used in the non-real-time data applications. Throughput measures how fast a certain amount of data is transferred through the network.

- Network QoS: Flow-based QoS measures can also be used to measure the quality of a network. Since, most of the time, the number of traffic flows in the network is very large, it is necessary to summarize each flow-based measure in one value, which is valid network-wide. Popular methods used are taking the average of flow-based measures (average packet loss rate) or using the extreme (maximum or minimum) values (maximum packet delay) of flow-based measures.

Often, it is too complex, or not possible at all, to determine flow-based QoS parameters for deriving averages or maximum points. In these cases, network QoS measures is required, based on traffic aggregates rather than individual flows. One such parameter, which relates traffic aggregates to QoS, is simply the average link utilization. It directly affects packet loss rates, queuing delay, and achievable throughput and, thus, can serve as an appropriate, link-specific QoS measure. In order to derive one network-wide QoS value, many different functions are meaningful **[9].** Most common utilization-based network QoS measures used in different network optimization studies are:

- The maximum link utilization of the network:

$$C_G = \max\left(\frac{UB(e)}{B(e)}\right); \quad e \in E : E \to R^+$$

  (This cost is evaluated by first routing all the demands and calculating the bandwidth requirements on each link and subtracting the link bandwidth values finally.)

- The maximum overloaded link utilization value of the network:

$$C_G = \max\left(\frac{UB(e) - B(e)}{B(e)}\right); UB(e) > B(e); \quad e \in E : E \to R^+$$

  (This cost is evaluated by first routing all the demands and calculating the bandwidth requirements on each link and subtracting the link bandwidth values finally.)

- The sum of the link overloading values of the network:

$$C_G = \sum_e (UB(e) - B(e)); \quad UB(e) > B(e), e \in E : E \to R^+$$

(This cost is evaluated by first routing all the demands and calculating the bandwidth requirements on each link and subtracting the link bandwidth values finally.)

- Hybrid utilization cost covering maximum link utilization and average link utilization, where maximum link utilization is multiplied by constant *a* to form a penalty function:

$$C_G = a * \max\left(\frac{UB(e)}{B(e)}\right) + \frac{\sum_e UB(e)/B(e)}{|E|}; \quad e \in E : E \to R^+$$

In the preceding section definition of a network is made and the main elements in mathematical forms are presented. Besides those presented elements there are some variables, which are frequently used in the network routing optimization problem. Every node in the network has its own traffic characteristics, i.e. demands and their requirements. There are also some other parameters used in realizing the routing of these demands, i.e. paths and related costs. Let's now define these necessary parameters, which are used frequently in the problem definition. Let:

- $r(s,d)$ be the demand (request) from source node $s \in N$ to destination node $d \in N$,
- $\Delta D_{s,d}$ be the delay constraint for demand $r(s,d)$
- $\Delta B_{s,d}$ be the bandwidth constraint for demand $r(s,d)$
- $p(s,d)$ be the path used for routing the demand $r(s,d)$ from node $s \in N$ to $d \in N$
- C_G be the cost of the network $G(N,E)$, which is basically the sum of costs of all the links $e(e \in E)$. This cost may be either a monetary cost or any measure of resource utilization:
- $td\{p(s,d)\}$ be the total delay of the path $p(s,d)$, which is sum of all link delays along the path $p(s,d)$:

$$td\{p(s,d)\} = \sum_{e \in p(s,d)} D(e)$$

- $bb\{p(s,d)\}$ be the bottleneck bandwidth of the path $p(s,d)$, which is minimum of the residual bandwidth of the links along the path $p(s,d)$:

$$bb\{p(s,d)\} = \min\{RB(e), e \in p(s,d)\}$$

So, given the communication network $G(N,E)$ and the demands, the network routing optimization problem is defined as:

"Given a fully defined network and its instantaneous traffic properties (demands matrix), find a routing solution, which minimizes the network objective function subject to certain QoS measures".

Minimize:	$C_G : E \rightarrow R^+$
Subject to:	$td\{p(s,d)\} \leq \Delta D_{s,d}; \forall (s,d) \in N$
	$bb\{p(s,d)\} \geq \Delta B_{s,d}; \forall (s,d) \in N$

The first constraint given is the delay constraint of any given demand, and the second constraint is the bandwidth requirement constraint of any demand.

In the next section, the routing optimization problem and the implementation considerations are related.

3.4 Implementation Issues of the Network Routing Optimization Problem of this Study

The general network model considered in this work, $G(N,E)$, is a set of $n = |N|$ switching nodes connected by $l = |E|$ physical links. Each of source-destination pairs $(s,d) : s,d \in G(N,E)$ (each request is denoted as $r(s,d)$) is connected by one logical link called a path, $p(s,d)$, which is created by sharing the capacities of some of the physical links connecting the nodes. A path carries a flow, $r(s,d)$, (the traffic demand of a source-destination pair initiated from the source to the destination) and the number of flow paths used at an instant is $n \times (n-1)$. All possible and feasible combinations of physical links that can make up a path for source-destination pairs must be taken into account.

In a dynamic routing problem, the network model is constant and only the traffic demand matrix and the constraints are changing. All possible paths for each source-destination pair can therefore be searched thoroughly in advance and arranged in a routing table in order of length with delay constraint.

A network with $n = |N|$ switching nodes forms $n \times (n-1)$ source-destination pairs $(s,d) : s,d \in G(N,E)$, where each (s,d) has $ncp(s,d) \in R^+ : s,d \in G(N,E)$ number of candidate flow paths formed according to the connection properties of the whole network, $G(N,E)$.

There are at least $\prod_{\forall r(s,d)} ncp(s,d)$ number of routing combinations for the general network. In real life, a communication network consists of 10, 20, or maybe more nodes. Considering such a huge network, solution space for the routing tables becomes very huge.

A network configuration determines the paths for all flows and, therefore, the capacity usage of each physical link. When the traffic demand for a physical link exceeds the capacity of the link, the fraction of the exceeded capacity or the traffic the link cannot carry is called the demand loss rate. Given a network and a demands matrix of all source-destination pairs, finding the network configuration that minimizes the demand loss rate should solve the routing problem. The solution space for this kind of optimization problem is large even for a small network.

The most important performance indicator of a communication network is the link utilization table. The idea in routing optimization is the minimization of maximum link utilization entry in the link utilization table, which means that as few packets as possible will be lost at routers' link interfaces and the overall delays will be minimized. Thus, naturally, the objective of the routing optimization problem is the minimization of the maximum link utilization in the network under given traffic demands matrix. Let's remember the routing optimization problem as defined in previous section:

"Given a fully defined network and its instantaneous traffic properties (demands matrix), find a routing solution, which minimizes the network objective function subject to certain QoS measures"

The objective function used in this study is a hybrid cost function including demand loss rate (DLR), average and maximum link utilization values as parameters. Demand loss rate is defined as the ratio of blocked demand over total demand. Whenever demand loss rate is

bigger than 0, the optimizer searches for minimizing the demand loss rate first because of the large penalty multiplier. Whenever the demand loss rate is decreased to a very small value, the optimizer tries to minimize maximum and average utilization values.

$$C_G = a * DLR + \max(\frac{UB(e) - B(e)}{B(e)}) + \frac{\sum_e (\frac{UB(e)}{B(e)})}{|E|}; \quad e \in E : E \to R^+$$

The bandwidth constraint of each demand is to be optimized through minimum DLR as seen in the objective function. The delay constraint of each demand is guaranteed through selection of the feasible paths. The optimizer shall, thus, try to find the least cost network configuration through set of delay constrained feasible paths for each flow. As long as the number of un-handled demands increases because of delay constraint, the objective function will linearly increase because of the DLR penalty. Thus, objective function need not include a delay parameter since all demands are handled in delay constraint boundary. This network routing optimization approach is called delay-constrained least cost routing, which is of NP-complete complexity but may be solved by different heuristics. As long as the definition of routing problem is made, the constraints on the network model are determined and the objective function for the routing optimization problem is made, it is now time to go further on discussing network routing optimization methods.

CHAPTER 4

OPTIMIZATION HEURISTICS FOR NETWORK ROUTING PROBLEM

4.1 Introduction

As a block of dynamic routing control, optimizer is used to adjust network routing tables or administrative link weight tables in order to use the available network resources as effective as possible. As discussed in the first chapter, dynamic routing control may choose a routing strategy between two alternatives: destination-based or flow-based routing. According to the routing strategy to be used, optimization methods that are available may change and implementations may vary in dynamic routing control. The network routing problem optimization methods can be classified as:

- Weight system dependent optimization methods (must use destination-based routing)
- K-routes system dependent optimization methods (must use flow-based routing)

Weight system dependent optimization methods are used to fix a weight system among the network, which indirectly helps determining all the shortest paths, and consequently, the points where traffic flows are split to their destinations, not allowing for any flexibility during network operation. It has been mentioned that, destination-based methods follow the next hop determination procedure, which is the basis of shortest path routing algorithms. Thus, an analogy between the weight system dependent routing methods and the destination-based routing strategies can be made. The weight system dependent optimization heuristics will be discussed in the following sub-sections.

On the other hand, *K-routes system dependent optimization methods* are used to choose the best set of routes from the fixed global routing table, which includes at most K predefined routes for each flow. If K-routes system global routing tables are examined, it can be seen that a network routing table is formed in such a way that each source-destination pair has maximum number of K route alternatives, arranged in shortest path first sequence. K-routes optimization methods naturally choose to use a global routing table, which does not include all routes for each node, or equally which is not equal to the solution space but is a subset of it. Working with a subset of solution space may result in failing to find the optimum point. Working with a large K may help us finding the optimum point but can easily slow down the optimizer. Each network must be examined carefully to choose a proper K because of the

trade off between optimum point convergence and the optimizer speed. An analogy between the K-routes optimization methods and the flow-based routing strategies can be made. On the other hand, there exists another optimization method, adaptive K-routes system dependent optimization method, which is hybrid in nature. Adaptive K-routes system dependent optimization methods can use both weight and K-routes system dependent optimization algorithms, where fast convergence of K-routes (flow based optimization) and strength of weight system dependent methods in reconstructing the K-routes global routing table according to the current load and link characteristics do exist. Adaptive K-routes system dependent methods always start with fixing link weights to under-load the network to a reasonable value, and then use the K-routes system to find the optimum point for the cost minimization problem. Changing the K-routes global routing table dynamically through weight system changes may solve the problem that arises when K is not enough to find an optimum point. Dynamic routing control, using adaptive K-routes optimization method works just like the one using K-routes optimization method. Both methods update the routing tables of each router in the communication network. Although, adaptive K-routes optimization method uses weight adjustment, it has nothing to do with updating the administrative link weight tables in the routers. To sum up, weight adjustment is only a local process in the optimizer subsystem of the central management unit.

In the next sections, two different weight system dependent optimization heuristics and two different genetic algorithms, which are weight and K-routes system adaptive respectively, will be discussed:

- Weight system dependent optimization methods:
    - Weights adjustment heuristic **[11]**,
    - Simulated annealing heuristic **[11]**,
    - Genetic algorithm (weight system encoding)**[11, 8, 13]** and parallel GA.
- K-routes system dependent optimization methods:
    - Genetic algorithm (K-routes encoding) **[1, 12]**.

4.2 Weights Adjustment Heuristic

Weights adjustment heuristic is a local search method, which tries to directly compute a feasible link weight system. The method iteratively adjusts the links' weights on the basis of the current link loads: in consecutive steps the algorithm increases the weights of overloaded links and decreases the weights of under-loaded ones [11]. This heuristic is successful in only minimizing the data loss on the links because of insufficient bandwidth. However, this heuristic may not make a good load balancing throughout the network because of its insufficient capabilities.

A modified version of weights adjustment heuristic that is proposed in [11] is used in this study. The algorithm used works according to the pseudo-code given in table 1:

Table 1: Weights adjustment pseudo-code.

```
1.  begin
2.   current_cost=evaluate_network_cost(current_weight_system);
3.   best_weight_system=current_weight_system;
4.   best_cost=current_cost;
5.   repeat
6.    if(current_cost<=best_cost)
7.     begin
8.      best_weight_system=current_weight_system;
9.      best_cost=current_cost;
10.    end
11.   for each link in the network do
12.    if(link is overloaded)
13.     link_weight++;
14.    else if(link is under_loaded)
15.     link_weight-=random(2);
16.   end for
17.   dijkstra_find_all_paths(current_weight_system);
18.   current_cost=evaluate_network_cost(current_weight_system);
19.  until(epoch==0 || best_cost<=BEST_COST_THRESHOLD)
20.  return best_weight_system & best_cost;
21. end
```

The algorithm starts from an initial weight system: *current_weight_system*. All demands are routed according to *current_weight_system*, and then the cost of the network *current_cost* is calculated according to the objective function of the optimizer. The best results of the optimizer *best_cost*, *best_weight_system* are initially made equal to *current_cost* and *current_weight_system* respectively. The algorithm starts with the *current_weight_system*

and at each turn it adjusts the link weights until a proper result is found or the epoch is expired. At each turn of the algorithm, if *current_cost* is found to be less than or equal to the *best_cost* then *best_cost* and *best_weight_system* are made equal to *current_cost* and *current_weight_system* respectively.

Weights adjustment heuristic relies on a mechanism, which adjusts the link weights not only randomly but also reasonably. Weights adjustment phase works as follows:

- If link $e(u,v); u,v \in N, e \in E)$ is overloaded, i.e. $UB(e) > B(e)$, then link weight w_e is increased by one. This decreases the chance that this link is used in the paths more.
- If link $e(u,v); u,v \in N, e \in E)$ is under-loaded, i.e. $UB(e) < B(e)$, then link weight w_e is decreased by a random value. This increases the chance that this link is used in the paths more.

The objective function of the algorithm can reflect one of three possible objectives: maximization of average free capacity, maximization of total free capacity, or minimization of the variance of the links' free capacity. To sum up, if one of the costs mentioned hopefully decreases through weights adjustment process (so the algorithm moves in the right direction), then the new weight system is accepted. Weights adjustment heuristic may not always find the optimum routing solution given a routing optimization problem because of its destination-based routing nature. But whenever an optimum point is available through destination-based routing, this heuristic works fine and fast.

4.3 Simulated Annealing Heuristic

SAN is a well-known multi-purpose meta-heuristic for combinatorial optimization. For some problems SAN is able to find solutions close to global optima, even for the problems with large solution spaces. The advantages of this heuristic are its general usability, easy adaptation to a particular application and easy implementation. This method is used in integer weight systems. [11]

The algorithm works according to the pseudo-code given in table 2:

Table 2: Simulated annealing pseudo-code, [11].

```
1.    Begin
2.     current_cost=evaluate_network_cost(current_weight_system);
3.     new_weight_system=current_weight_system;
4.     best_weight_system=current_weight_system;
5.     new_cost=current_cost;
6.     best_cost=current_cost;
7.     initialize(current_temperature, temperature_lower_bound, counter,
8.    counter_upper_bound,
9.                 temperature_decrease_factor, delta_cost);
10.    repeat
11.     for counter:=0 to counter_upper_bound do
12.      new_weight_system=neighbor(current_neighbour_system);
13.      dijkstra_find_all_paths(new_weight_system);
14.      new_cost= evaluate_network_cost(new_weight_system);
15.      delta_cost=new_cost-current_cost;
16.      if(deltaCost<=0)
17.       begin
18.        current_cost=new_cost;
19.        current_weight_system=new_weight_system;
20.        if(current_cost<=best_cost)
21.         begin
22.          best_cost=current_cost;
23.          best_weight_system=current_weight_system;
24.         end
25.       end
26.      else /*else of if(deltaCost<=0)*/
27.       begin
28.        random=random(10000)/10000;
29.        if(random< exp{- ΔC /T}) /*METROPOLIS TEST*/
30.         begin
31.          current_cost=new_cost;
32.          current_weight_system=new_weight_system;
33.         end
34.       end
35.     end for
36.     current_temperature=current_temperature* temperature_decrease_factor;
37.    until(current_temperature<temperature_lower_bound ||
38.          best_cost<cost_lower_bound);
39.    return best_weight_system & best_cost;
40.   end
```

Starting with solution *current_weight_system*, all the demands are routed accordingly. Then at each step, the algorithm selects a random neighbor of current_weight_system, *new_weight_system*, using neighbor() function. The neighboring state *new_weight_system*, is obtained by selecting at random links and incrementing or decrementing the weights by 1 (the selection from the two possibilities is also random). Then the demands are routed according to the *new_weight_system* and the cost of the new state, *new_cost*, is calculated and compared to *current_cost*. If the cost of the new state is not greater than of the old one, the new state is always accepted. If *new_cost* is greater than *current_cost*, the new state is accepted according to the Metropolis test (see table 2, lines 28 to 33). The outcome of the test depends on the *current_temperature* (basic control parameter of SAN) and on the cost difference between the states, *delta_cost*. At the beginning, SAN will accept states with relatively large *delta_cost* with a high probability, which is then decreased exponentially during the optimization process. This allows for a better scanning of the state space to avoid local optimum points. At the end of the main loop the current_temperature is decreased by *temperature_decrease_factor* (*temperature_decrease_factor*<1); and the loop is executed until the temperature reaches a predefined lower bound. The objective function of the algorithm can reflect one of three possible objectives: maximization of average free capacity, maximization of total free capacity, or minimization of the variance of the links' free capacity. The best cost and system ever achieved, *best_weight_system* and *best_cost* are returned. **[11]**

4.4 Genetic Algorithms

In this section an overview of genetic algorithms will be made, **[8, 11, 12, 13]**, and two different genetic algorithms will be described in the next two sections.

The idea of using an evolutionary mechanism for search and optimization purposes was first put forward by J.H. Holland in 1975. The evolutionary algorithm, so called Genetic Algorithm, since then, has found so many implementation areas in optimization world. Genetic algorithms aim to find the global optimum point for an objective function, while still fulfilling the constraints of the given optimization problem. Besides, this algorithm can be robustly applied to any optimization problem, because of its talent for avoiding local-optima traps through randomization and mutation processes.

Evolution of genes through each generation continues thanks to natural selection, reproduction and mutation processes. Natural selection is the common idea that all genetic algorithms are based on. The strength of an individual to survive in the population depends on the gene structure he has. According to the natural selection principles, the probability that a good gene will survive and prevail in the next generation is much more higher than the probability that a bad gene will survive. Since the number of individuals having good gene combinations will be more through natural selection, reproduction will take place between the good individuals, and the number of bad genes in any individual's gene string will decrease in next generations. Besides, each generation will experience a mutation because of environmental factors and this will increase the probability of having fit, robust or durable individuals in the next generations.

Each combination of genes is called as an individual or chromosome in genetic algorithms, and these are the possible solutions for the optimization problem. Individuals come together and form a population and through natural selection, reproduction and mutation processes good genes are inherited to the next generations.

Application of genetic algorithms to optimization problems must be handled with care considering genetic material encoding and gene string definitions. The representation of an individual in genetic algorithms is of great importance and plays an important role in the speed and success of the algorithm. Each individual must be encoded carefully such that his gene combination can be transformed into a valid solution for evolutionary processes. With every gene string, a value, fitness value, is associated to that individual, which quantifies the quality of the solution. These fitness values are used mainly in natural selection process. Let us now go through natural selection, reproduction and mutation processes:

- Natural Selection: The new generation can only be reproduced by bringing together the good individuals according to the fitness values they have. All individuals in the current generation are sorted according to their fitness values and the good ones are selected to survive where the bad ones pass away without having any chance to prevail in the next generations.
- Reproduction: Reproduction is the process of creating a new generation by using the individuals that survived through natural selection mechanism. Reproduction process continues until a predefined population size is reached. This is accomplished by crossing-over the individuals that survived in natural selection. The new offsprings have exactly the same gene string fragments of their parents.

- Mutation: Mutation can be thought of an associative process to reproduction. It is the process of introducing new genetic materials to the individuals in the population. Thus every new generation can introduce new solutions to the problem, and dominance of some fittest gene strings over new generations is avoided. Mutation is needed, because, after some number of generations, the number of different strings decreases because of the dominance of the fittest individuals and reproduction phase results in cloned generations over and over. Thus, mutation process shifts the optimum point search to new locations in the solution space and randomly alters the genes.

The sequence of the mentioned operators is shown in figure 3:

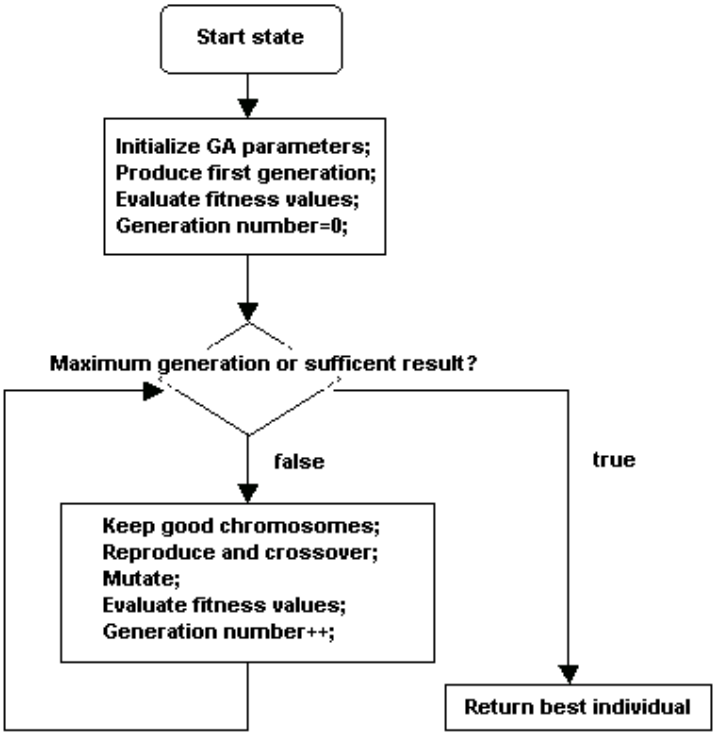


Figure 3: Genetic algorithm state flow chart.

Starting with an initial random generation, applying natural selection, reproduction, crossover and mutation over and over until reaching the maximum number of generations permitted or the minimum cost, a minimum cost routing solution can be found. Every string

in the initial generation is labeled with a fitness value showing the string's quality to survive. The strongest, fittest strings are chosen to perform crossover operation on them. A mutation process follows the crossover operation, and all new strings are shifted to new locations in the solution space. This results in a new generation where all mentioned processes are applied again and again until maximum number of generations is produced. Finally, as the maximum number is reached, the fittest individual (best solution, string) is selected and presented as the optimum point for the solution space.

As discussed before, every genetic algorithm is represented and distinguished by the encoding methodology it uses. Encoding methodology is the representation of the elements of the solution space as strings of genes, which is also called the string representation. The structure of the genetic material that the encoding methodology uses must allow representation of every possible solution in the solution space. Another important appropriateness constraint for the encoding methodology is that the strings modified during crossover and mutation operators again must be the elements of the solution space. Network routing optimization problems may define the solution space as a specific weighted link topology, a routing table or sometimes a capacity allocation scheme. There are two main encoding schemes used in network routing optimization problems:

- Weight system encoding
- K-routes system encoding.

These encoding schemes and the corresponding algorithms will be discussed in the next two sections.

4.4.1 Weight System Dependent Genetic Algorithm and Parallelization of the Algorithm

Weight system dependent genetic algorithms are those routing methods classified in the destination-based routing concept. In the papers [8], [9], [13], [14], [20] different implementations of weight system dependent genetic algorithms are proposed to solve network routing optimization problem. In this section, the structure of weight system dependent genetic algorithms is explained and the fastest weight system dependent genetic algorithm proposed in [14] is modeled.

Let's return back to the definition of a network: "A network is modeled as a directed and connected graph $G(N, E)$ where N is a finite set of vertices (network nodes) and E is the

set of edges (network links) connecting these vertices". Each link, $e(u,v)$ connecting nodes u and v has bandwidth $B(e): E \to R^+$ and delay $D(e): E \to R^+$ characteristics, which are used actively in routing algorithms such as OSPF and EIGRP. Each link $e(u,v)$ has an administrative weight $w(u,v): W \to R^+$ and a delay $d(u,v): D \to R^+$ associated to it. This administrative weight is main parameter being used in weight system dependent genetic algorithms. Each individual is encoded as a string of weights representing each link in the solution space and used for forming a shortest paths routing table for the given network. Shortest path for each demand is found according to the delay constraint of that demand. The path used for routing a demand should obey the delay constraint of that demand. Thus total delay of a path should not exceed the delay constraint. Let's remember the delay constraint of the network routing problem:

$$td\{p(s,d)\} \le \Delta D_{s,d}; \forall (s,d) \in N .$$

Thus, resulting routing table is naturally a solution candidate, which is fulfilling each demand's delay constraints. Routing table is used to distribute loads and a cost is evaluated for that chromosome. Fitness value of the chromosome is determined by the cost evaluated. Figure 4 shows the encoding scheme and the fitness calculation process in the order of encoding, routing and cost calculation.

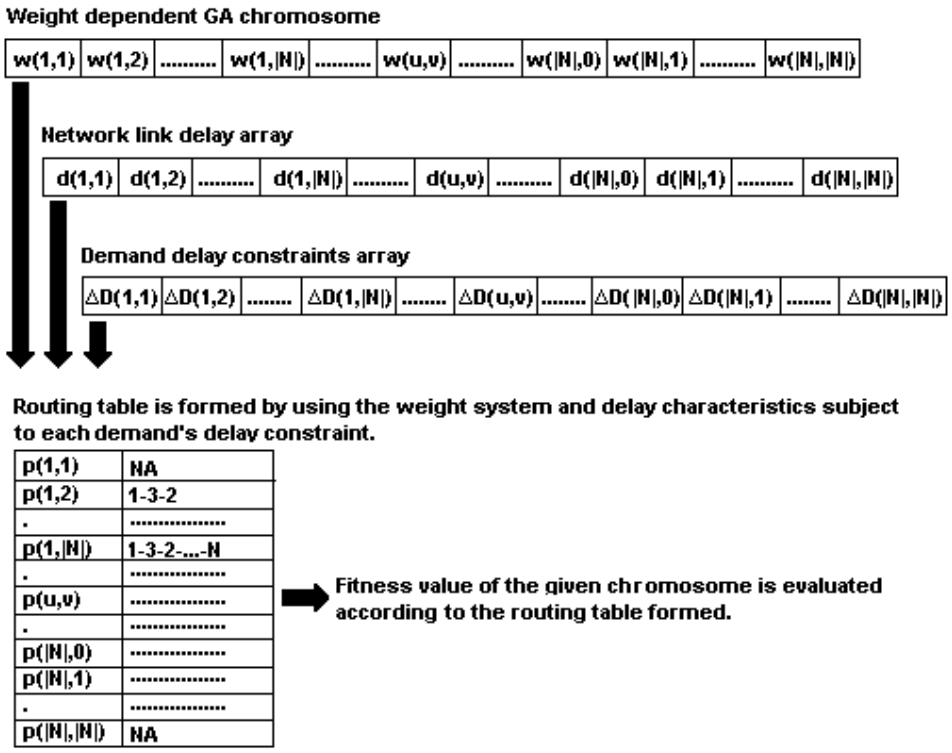


Figure 4: Weight system dependent GA chromosome structure.

As told before, every genetic algorithm starts by forming an initial generation randomly. This generation may include individuals representing current weight system and some individuals of past solutions. This is called the re-use of past solutions and is introduced as an improvement on the classical genetic algorithms [1]. This approach can be used in genetic algorithm implementations to speed up the process and to converge to the optimal point faster. But this procedure must be handled with care [9]:

"However, it has been observed that if at the beginning a solution is injected, which is greatly superior over other randomly selected strings, the algorithm often gets stuck there and does not find a better solution. The reason for this is that the injected string would have such a high fitness value that other strings hardly have a chance to survive at all. Within only a few generations, the randomly selected gene strings would die out and only the initially injected solution would be left over. Therefore, presetting strings should be done with care, and it should only be one trial out of many." [9]

In this study, formation of the initial population is made through a wiser approach rather than a randomized formation. Formation of each generation in the evaluation flow includes this approach also. We propose another method where we use the idea behind the weights adjustment heuristic. Initial population's first individual is always the last weight system that network works on. Creation of the other individuals is done according to the following process:

- Clone current weight system as the first individual.
- Determine all the overloaded and under loaded links according to the first individual of the population (current weight system).
- Fill the population with randomly generated individuals.
- Mutate each of the randomly generated individuals according to the following rules:
    - For each of the under loaded links, decrease the weight of that link by a random value. Recall that each gene in the individual string corresponds to weight of a link.
    - For each of the overloaded links, increase the weight of that link by a random value.

In [8], the writer proposed a hybrid genetic algorithm where at the beginning of each generation formation, a local search heuristic is executed to shape the individuals according to the network conditions (see figure 5). This method is a little bit different than the approach we follow. First the initial generation is formed randomly in [8], and second the best individual of the population is selected and modified in [8]. Let's think about the second difference. As told before, injecting superior solution candidates into the population may lead the algorithm to be stuck. Thus, performing a local search should be handled with care. Evolution process includes a modification in this study also.

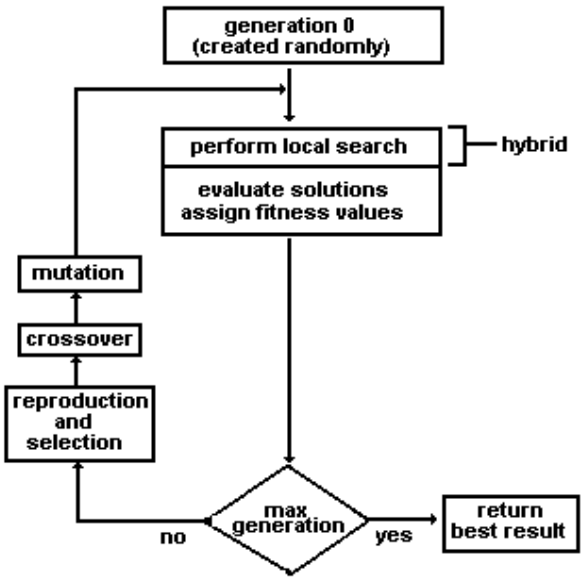


Figure 5: Hybrid GA [8]

Genetic algorithm goes on with the evolution process after initializing the very first generation. In each epoch of the genetic algorithm, a new generation is introduced for genetic operations. Each generation, is first subjected to fitness value evaluation process to enter natural selection. A quick sort algorithm is used to sort individuals according to their fitness values. There are two selection mechanisms applied on the population. First selection is applied to remove some of the least fit individuals from the population. The second selection is applied to select the parent chromosomes for crossover. Parents' selection is based on rank selection implementation to make the probability to be selected a little bit more balanced for all chromosomes in the population. Selection process is simply dividing the current population to subsets as parents, survivors and lost ones. Figure 6 shows the population dynamics of this algorithm. **[14]**

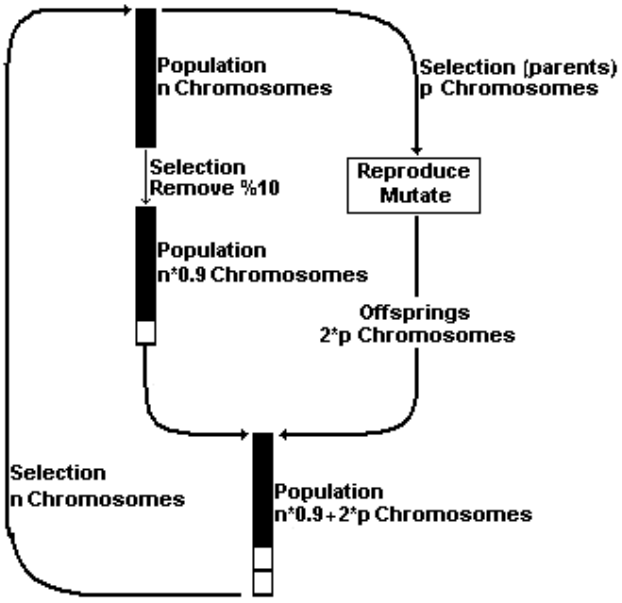


Figure 6: Population dynamics for weight system dependent GA, [14].

The combined reproduction and mutation process follows the selection process. The population, which is smaller than its normal size is now to be filled with offsprings that are reproduced by crossing over selected parents and mutation. During generation of a new chromosome, a random number is generated for each gene and according to the outcome each gene is inherited from one of the parents (crossover) or formed randomly (mutation). In the hybrid genetic algorithm, %50 of the new individuals generated is processed through another mutation process like the one in the starting population generation. Those selected individuals are mutated according to the following rules:

- Determine all the overloaded and under loaded links according to the best individual of the population.
- Mutate selected individual according to the following rules:
    - For each of the under loaded links, decrease the weight of that link by a random value.
    - For each of the overloaded links, increase the weight of that link by a random value.

Mutation and reproduction thresholds are used to decide on the action in this process. Mutation threshold value is 3% and reproduction threshold value is 53% for a good convergence of the algorithm and these values have been tested in the algorithm

implementation. Table 3 includes the pseudo-code for combined reproduction and mutation process.

Table 3: Combined reproduction and mutation process.

1. for all genes $w_k(u,v): W \rightarrow R^+$ of the offspring **do**
2. generate $r = random(0,100)$
3. if ($r \langle MutationThreshold$)
4. $w_k = random(0, MaxWeight)$
5. else if ($r \langle \text{Re}\, productionThreshold$)
6. $w_k = w_k^{parent1}$
7. else
8. $w_k = w_k^{parent2}$
9. end for

Genetic algorithms depending on weight system encoding usually work slower than the genetic algorithms using K-routes encoding. Each individual requires a reconstruction of shortest path routing table according to the encoded weight system. Regarding weight system dependent optimization methods, genetic algorithms depending on weight system encoding cannot work as fast as the weights adjustment heuristic, which has been mentioned in the previous section. However, weight adjustment heuristic generally cannot handle utilization optimization if network is under-loaded, thus GA(W) has an advantage over WA in delay constraint bandwidth optimization problems. The capabilities of genetic algorithms and their ease of use with any problem guide researchers to develop new methods for speeding up these algorithms. One of the most important developments in genetic algorithms is the adaptation of distributed computing on the genetic algorithms. There are some distributed genetic algorithm implementations where parallel computers share the fitness evaluation process in a distributed manner. Figure 7 shows the structure of the parallel genetic algorithm.

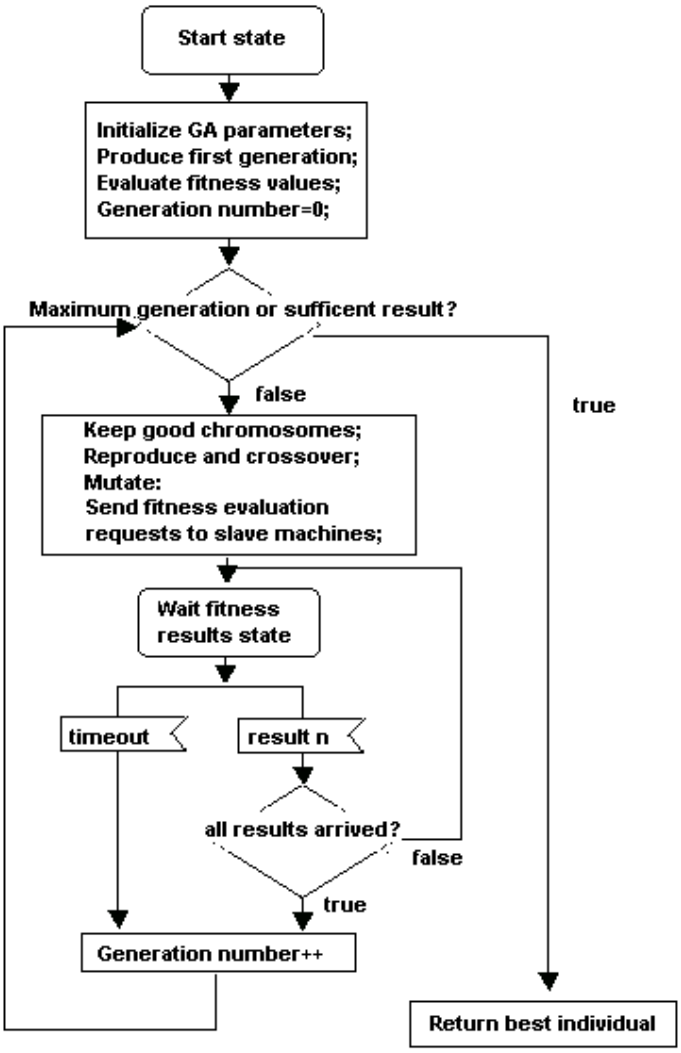

Start state
Initialize GA parameters;
Produce first generation;
Evaluate fitness values;
Generation number=0;
Maximum generation or sufficent result?
false
true
Keep good chromosomes;
Reproduce and crossover;
Mutate:
Send fitness evaluation
requests to slave machines;
Wait fitness
results state
timeout
result n
all results arrived?
false
true
Generation number++
Return best individual

Figure 7: PGA

A server computer running the main genetic algorithm processes asks some slave computers to evaluate each individual's fitness value in parallel. In some cases, evaluation of each individual's fitness values sequentially on one machine takes more time than evaluating the fitness values of different individuals on parallel slave machines. When the number of individuals or the network size become larger, distributed fitness evaluation makes the genetic algorithm faster. But in the case of a small network, or small population size, distributed fitness evaluation may take more time because of communication latencies between parallel machines.

In this section, weight system dependent genetic algorithm was discussed and implementation related information was given. In the next section, K-routes system dependent genetic algorithm will be discussed.

4.4.2 K-routes System Dependent Genetic Algorithm

K-routes system dependent genetic algorithms are those routing methods classified in the flow-based routing concept. In the papers [1], [12], [15] different implementations of K-routes system dependent genetic algorithms are proposed to solve network routing optimization problem. In this section, the structure of K-routes system dependent genetic algorithms is explained and the algorithm implementation is modeled.

K-routes system dependent genetic algorithm works with a global routing table, which includes a maximum number of K alternative paths for each possible flow in the communication network. Each pair of nodes in the network is connected by a logical link called a path, which is created by sharing the capacities of some of the physical links connecting the pair of nodes. There are usually many possible combinations of physical links that can make up a path between any two nodes. These possible combinations of physical links are called possible routes for the flow and a set of the selected routes is called a network configuration. On the other hand, the K-routes global routing table where the selection of routes for network configuration is made includes a maximum number of K different route alternatives for each flow. The aim of K-routes system dependent genetic algorithm is to build a network configuration from the K-routes global routing table to utilize the communication network better. K-routes system dependent genetic algorithm uses an encoding mechanism where each of the genes in a chromosome represents the alternative route number to be used for a flow. Encoding mechanism is explained in figure 8:

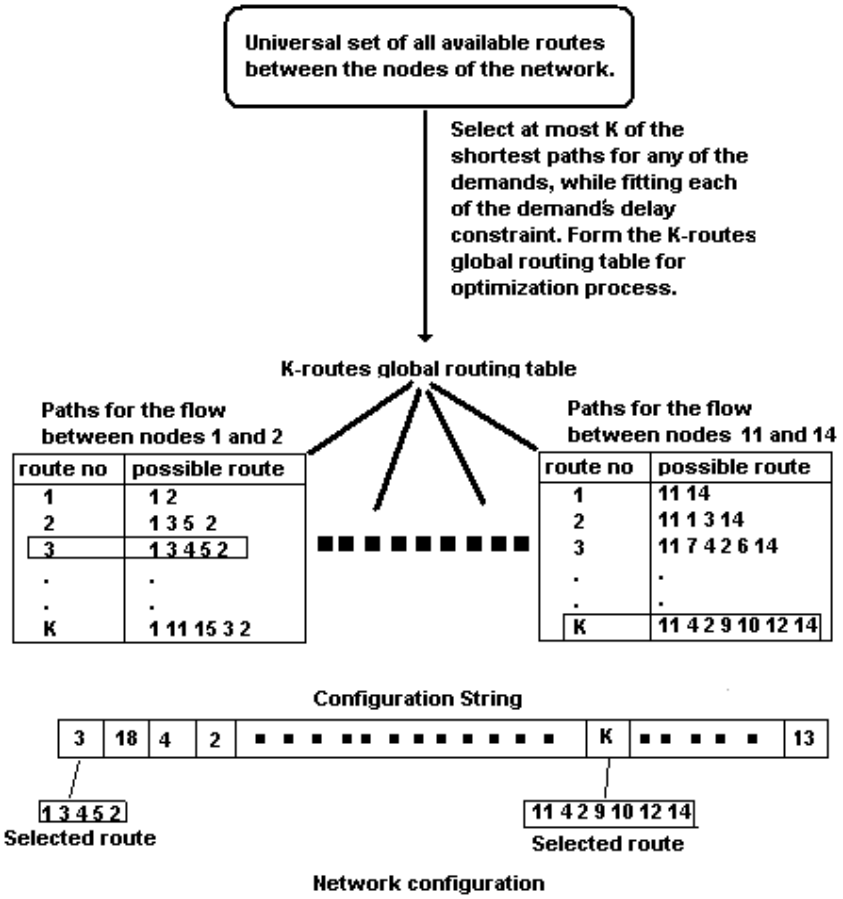


Figure 8: K-routes encoding.

As seen in figure 8, there exists a routing table for each flow in the communication network. There are at most K number of alternative routes to cover the corresponding demand. The alternative routes that are used in optimization process are chosen such that each of the selected routes for a demand fits the delay constraint of the demand. There can be less than K alternative routes for a flow in the routing table. Delay constraint of each flow may limit the number of available paths. The configuration string in the figure represents one of the individuals in the genetic algorithm encoded in K-routes system. Each gene of the configuration string corresponds to a route number realizing the corresponding flow.

K-routes system dependent genetic algorithm works like the weight system dependent genetic algorithm, but with a difference in the encoding mechanism. K-routes system

dependent genetic algorithm works faster than the weight system dependent genetic algorithm since it does not need to reconstruct routing tables for each of the individuals. But as mentioned before, selection of the number K may result in drastic changes in the performance of the optimizer. Selecting a small K has the possibility that the algorithm may not reach the optimum point since the global routing table may not include optimum set of routes inside. Selecting a large K may result in a decrease in the speed of the algorithm since the number of individual gene combinations increases by increasing K.

K-routes genetic algorithm always starts with a randomly generated initial population. Genetic algorithm goes on with the evolution process after initializing the very first generation. In each epoch of the genetic algorithm, a new generation is introduced for genetic operations. Each generation, is first subjected to fitness value evaluation process to enter natural selection. A quick sort algorithm is used to sort individuals according to their fitness values. K-routes genetic algorithm implementation of this study does not include the rank selection method used in the weight system dependent genetic algorithm. Instead, a number of the best individuals are selected for the next generation and reproduction. This is called the parents' selection phase. Combined reproduction and mutation process is also applied in the K-routes genetic algorithm implementation. Each offspring is reproduced from the parents selected before. Mutation is also applied online in the reproduction phase. Mutation and reproduction thresholds are used to decide on the action in this process. Mutation threshold value is 3% and reproduction threshold value is 53% as they were in the weight system dependent genetic algorithm implementation. Table 4 shows the combined reproduction and selection process.

Table 4: Combined reproduction and mutation process in K-routes GA.

1. for all genes $r_k(u,v): r_k \le K$ of the offspring **do**
2. generate $rand = random(0,100)$
3. if ($rand \langle MutationThreshold$)
4. $r_k = random(0, MaxRouteNumber)$
5. else if ($rand \langle \mathrm{Re}\ productionThreshold$)
6. $r_k = r_k^{parent1}$
7. else
8. $r_k = r_k^{parent2}$
9. end for

Each generation is processed over and over until reaching a desired solution or the generation threshold. It has been observed that the algorithm sometimes gets stuck and cannot produce new individuals because of some superior parents. There is always a chance that some superior individuals force the algorithm to go away from the optimum point and get stuck in some local optimum traps for a long time. Mutation process always helps the algorithm to escape from these traps, but it may need some time to recover because of the low mutation rate. The number of superior individuals increases as a result of crossover process until mutation process finds a new direction for the evolution process. It can be obviously seen that in the K-routes encoding scheme the effect of the mutation on one-gene results in the change of only one path in the routing table. However, in the weight system decoding, mutation of one gene (one link weight) may result in several path changes in the routing table. Thus, mutation in weight system dependent genetic algorithm helps the algorithm to escape from local optimum points or traps more than the mutation applied in the K-routes system dependent genetic algorithm. The key point is that, in such a case the crossover process fills the K-routes encoded population with superior individuals more and more in successive generations. Thus, a modification in the parents' selection phase of the K-routes system dependent genetic algorithm is made. Whole population is first sorted according to the fitness values and then all individuals having the same fitness values are reduced to only one individual so that superior individuals cannot coexist. Thus, this elimination system gives a chance to some of the less-fit, perhaps mutated versions of previously deleted superior individuals to pass their genes to next generations.

K-routes algorithm, unlike weight system dependent genetic algorithm, lets the network use a larger set of path combinations. By this way, algorithm can be more successful in reaching a global optimum point in some network configurations with given traffic properties. However, current destination-based routing protocols seem more robust than flow based routing algorithms and traffic engineers tend to use weight system dependent routing optimization methods **[9]**.

In the following chapter, performance analysis of optimization heuristics through dynamic routing control simulation program (DRCSP) is told.

CHAPTER 5

PERFORMANCE ANALYSIS OF OPTIMIZATION HEURISTICS THROUGH DRCSP

In this chapter, an analysis of the optimization heuristics on two different networks will be made. The first network (see figure 9) has been chosen intentionally from [11] to compare the performance of the optimization heuristics in this work and [11]. The second network (see figure 10) is chosen to make an analysis of the optimization heuristics on a larger network.

All the experiments are made with hybrid cost function:

$$C_G = a * DLR + \max\left(\frac{UB(e) - B(e)}{B(e)}\right) + \frac{\sum_e \left(\frac{UB(e)}{B(e)}\right)}{|E|}; \quad e \in E : E \rightarrow R^+$$

Thus the optimization results will show how effective the heuristics are on preventing data loss and load balancing on communication links.

5.1 Network Configurations

Two different networks are used for the experiments "12-nodes network" and "30-nodes network". Weight systems of these networks are configured in such a way that all the links in the network have the same administrative link weights. Thus, both of the networks initially have a routing solution based on shortest paths regarding number of hops. The networks used in the performance analysis are as follows:

- 12-nodes network: The first map is taken from [11]. The network can be seen in figure 9, the link bandwidth values (in units/sec) in table 5 and link delay values in table 6. All the links on the network initially have equal weights, which, in fact, cause the routing tables to be built on shortest paths regarding number of hops with delay constraint of each demand.

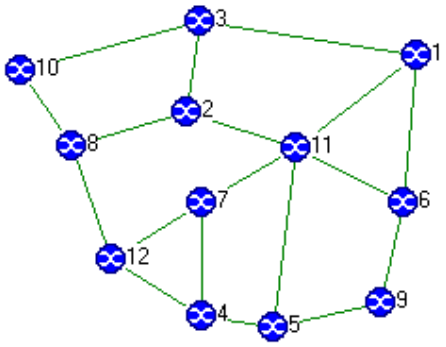


Figure 9: Network from [11].

Table 5: Link bandwidths for 12-nodes network (in units/sec).

B(1,3)=17	B(1,6)=26	B(1,11)=89
B(2,3)=273	B(2,8)=391	B(2,11)=545
B(3,10)=9		
B(4,5)=216	B(4,7)=146	B(4,12)=287
B(5,9)=122	B(5,11)=127	
B(6,9)=137	B(6,11)=242	
B(7,11)=284	B(7,12)=177	
B(8,10)=160	B(8,12)=328	

Table 6: Link delay values for 12-nodes network (in msec).

$$D(e) = 100; \forall e \in E$$

- 30-nodes network: The second network can be seen in figure 10, link bandwidth values (in units/sec) in table 7 and link delay values in table 8. All the links on the network initially have equal weights, which, in fact, cause the routing tables to be built on shortest paths regarding number of hops with delay constraint of each demand.

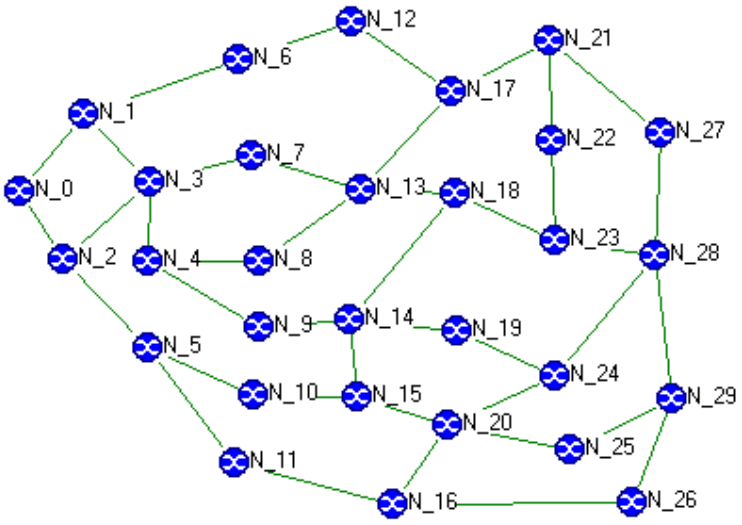


Figure 10: 30-nodes network.

Table 7: Link bandwidths for 30-nodes network (in units/sec).

$B(e) = 100; \forall e \in E$

Table 8: Link delay values for 30-nodes network (in msec).

$D(e) = 3; \forall e \in E$

5.2 Experiments

Performance analysis includes 3 experiments made on the mentioned network configurations by using DRCSP on a 1.7 GHz P4 machine. First experiment is made to compare heuristic performances on a 12-nodes network, which is configured such that all the links are fully utilized at the global optimum point of the problem. Second experiment is also made to show the heuristic characteristics but, on a 30-nodes network, which is configured such that a routing solution where all links are fully utilized exists in the solution space of flow-based routing technique. Last and the third of the experiments is made to show the performances of the optimization heuristics on a dynamical network where some selected nodes produce random demands destined for other nodes.

5.2.1 Experiment 1

The first experiment is made to compare the speed and the achievement of optimization heuristics to reach the global optimum point of the 12-nodes network configuration with the demands matrix to be given. This network configuration is taken from [11] and the results of the experiment will be compared with those of the experiment made in [11]. The demands matrix for 12-nodes network is chosen such that the optimum point for the routing optimization problem is a case that all links in the given network are fully loaded without any data loss [11].

It is known that, in the 12-nodes network there is a solution, which can be reached by both destination and flow based routing, in other words, by both weight and K-routes system dependent optimization heuristics. The cost function used in the simulation is the hybrid cost function mentioned in section 3.4:

$$C_G = a * DLR + \max\left(\frac{UB(e) - B(e)}{B(e)}\right) + \frac{\sum_e \left(\frac{UB(e)}{B(e)}\right)}{|E|}; \quad e \in E : E \to R^+$$

The delay constraint of each demand is 24 msec, and the demands matrix realized for 12-nodes network is given in table 9:

Table 9: Experiment 1, demands matrix for 12-nodes network (in units/sec).

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	10	11	13	13	16	7	19	10	6	13	14
2	0	0	3	21	19	21	14	32	14	5	68	32
3	0	0	0	27	21	25	17	35	14	3	107	24
4	0	0	0	0	11	21	15	51	21	19	84	40
5	0	0	0	0	0	13	20	35	5	18	74	28
6	0	0	0	0	0	0	16	34	4	9	30	18
7	0	0	0	0	0	0	0	21	14	12	4	15
8	0	0	0	0	0	0	0	0	28	47	129	14
9	0	0	0	0	0	0	0	0	0	7	61	13
10	0	0	0	0	0	0	0	0	0	0	24	19
11	0	0	0	0	0	0	0	0	0	0	0	97
12	0	0	0	0	0	0	0	0	0	0	0	0

Given the 12-nodes network configuration and the demands matrix, optimization heuristics were applied on the problem and results similar with those of [11] were reached. Table 10 shows that all optimization heuristics have reached the global optimum point where the network cost is 2. The global optimum point is the case where all links are fully loaded without any data loss. Thus, the cost, 2, comes from maximum link utilization, 1, plus average utilization, 1, and plus 0 DLR times a constant. Weights adjustment heuristic has been the fastest one among others, where SAN, GA(W) and GA(K) come after.

Table 10: Performance of optimization heuristics in experiment 1.

	NONE	WA	SAN	GA(W)	GA(K)
Time(msec)	-	≈50	≈600	≈1060	≈3000
Cost(units/sec)	178.3640	2	2	2	2

Weight system dependent optimization heuristics, WA, SAN and GA(W) have all resulted with the same network routing table. However K-routes system dependent optimization heuristic GA(K) has resulted with a different routing table. The routing tables before and after the optimization heuristics can be seen in table 11:

Table 11: Experiment 1, routing tables.

S-D	Initial	WA, SAN, GA(W)	GA(K)
1-2	1-11-2.	1-11-2.	1-11-2.
1-3	1-3.	1-3.	1-3.
1-4	1-11-7-4.	1-11-7-4.	1-11-7-4.
1-5	1-11-5.	1-11-5.	1-11-5.
1-6	1-6.	1-6.	1-6.
1-7	1-11-7.	1-11-7.	1-11-7.
1-8	1-3-10-8.	1-11-2-8.	1-11-2-8.
1-9	1-6-9.	1-6-9.	1-6-9.
1-10	1-3-10.	1-3-10.	1-3-10.
1-11	1-11.	1-11.	1-11.
1-12	1-11-7-12	1-11-7-12.	1-11-7-12.
2-1	2-11-1.	2-11-1.	2-11-1.
2-3	2-3.	2-3.	2-3.
2-4	2-8-12-4.	2-8-12-4.	2-8-12-4.
2-5	2-11-5.	2-11-5.	2-11-5.
2-6	2-11-6.	2-11-6.	2-11-6.
2-7	2-11-7.	2-11-7.	2-11-7.
2-8	2-8.	2-8.	2-8.
2-9	2-11-6-9.	2-11-6-9.	2-11-6-9.
2-10	2-8-10.	2-8-10.	2-8-10.
2-11	2-11.	2-11.	2-11.
2-12	2-8-12.	2-8-12.	2-8-12.
3-1	3-1.	3-1.	3-1.
3-2	3-2.	3-2.	3-2.
3-4	3-10-8-12-4.	3-2-8-12-4.	3-2-8-12-4.
3-5	3-2-11-5.	3-2-11-5.	3-2-11-5.
3-6	3-1-6.	3-2-11-6.	3-2-11-6.
3-7	3-2-11-7.	3-2-11-7.	3-2-11-7.
3-8	3-10-8.	3-2-8.	3-2-8.
3-9	3-1-6-9.	3-2-11-6-9.	3-2-11-6-9.
3-10	3-10.	3-10.	3-10.
3-11	3-2-11.	3-2-11.	3-2-11.
3-12	3-10-8-12.	3-2-8-12.	3-2-8-12.
4-1	4-7-11-1.	4-7-11-1.	4-7-11-1.
4-2	4-7-11-2.	4-12-8-2.	4-12-8-2.
4-3	4-12-8-10-3.	4-12-8-2-3.	4-12-8-10-3.
4-5	4-5.	4-5.	4-5.
4-6	4-7-11-6.	4-5-9-6.	4-5-9-6.
4-7	4-7.	4-7.	4-7.
4-8	4-12-8.	4-12-8.	4-12-8.
4-9	4-5-9.	4-5-9.	4-5-9.
4-10	4-12-8-10.	4-12-8-10.	4-12-8-10.
4-11	4-7-11.	4-7-11.	4-7-11.
4-12	4-12.	4-12.	4-12.
5-1	5-11-1.	5-11-1.	5-11-1.
5-2	5-11-2.	5-11-2.	5-11-2.
5-3	5-11-2-3.	5-11-2-3.	5-11-2-3.
5-4	5-4.	5-4.	5-4.
5-6	5-11-6.	5-9-6.	5-9-6.
5-7	5-11-7.	5-4-7.	5-4-7.
5-8	5-4-12-8.	5-4-12-8.	5-4-12-8.
5-9	5-9.	5-9.	5-9.
5-10	5-4-12-8-10.	5-4-12-8-10.	5-4-12-8-10.
5-11	5-11.	5-11.	5-11.
5-12	5-4-12.	5-4-12.	5-4-12.
6-1	6-1.	6-1.	6-1.
6-2	6-11-2.	6-11-2.	6-11-2.
6-3	6-1-3.	6-11-2-3.	6-1-3.
6-4	6-11-7-4.	6-9-5-4.	6-11-7-4.
6-5	6-11-5.	6-9-5.	6-11-5.
6-7	6-11-7.	6-11-7.	6-11-7.
6-8	6-11-2-8.	6-11-2-8.	6-11-2-8.
6-9	6-9.	6-9.	6-9.
6-10	6-1-3-10.	6-11-2-8-10.	6-11-2-8-10.
6-11	6-11.	6-11.	6-11.
6-12	6-11-7-12.	6-11-7-12.	6-11-7-12.

Table 11: Experiment 1, routing tables (cont'd).

S-D	Initial	WA, SAN, GA(W)	GA(K)
7-1	7-11-2.	7-11-2.	7-11-2.
7-2	7-11-2-3.	7-11-2-3.	7-11-2-3.
7-3	7-4.	7-4.	7-4.
7-4	7-11-5.	7-4-5.	7-11-5.
7-5	7-11-6.	7-11-6.	7-11-6.
7-6	7-12-8.	7-12-8.	7-12-8.
7-8	7-11-6-9.	7-4-5-9.	7-4-5-9.
7-9	7-12-8-10.	7-12-8-10.	7-12-8-10.
7-10	7-11.	7-11.	7-11.
7-11	7-12.	7-12.	7-12.
7-12	8-2-11-1.	8-2-11-1.	8-10-3-1.
8-1	8-2.	8-2.	8-2.
8-2	8-10-3.	8-2-3.	8-10-3.
8-3	8-12-4.	8-12-4.	8-12-4.
8-4	8-2-11-5.	8-12-4-5.	8-12-4-5.
8-5	8-2-11-6.	8-2-11-6.	8-2-11-6.
8-6	8-12-7.	8-12-7.	8-12-7.
8-7	8-2-11-6-9.	8-12-4-5-9.	8-12-4-5-9.
8-9	8-10.	8-10.	8-10.
8-10	8-2-11.	8-2-11.	8-2-11.
8-11	8-12.	8-12.	8-12.
8-12	9-6-1.	9-6-1.	9-6-1.
9-1	9-6-11-2.	9-6-11-2.	9-6-11-2.
9-2	9-6-1-3.	9-6-11-2-3.	9-6-1-3.
9-3	9-5-4.	9-5-4.	9-5-4.
9-4	9-5.	9-5.	9-5.
9-5	9-6.	9-6.	9-6.
9-6	9-6-11-7.	9-5-4-7.	9-6-11-7.
9-7	9-5-4-12-8.	9-5-4-12-8.	9-6-11-2-8.
9-8	9-6-1-3-10.	9-5-4-12-8-10.	9-5-4-12-8-10.
9-10	9-6-11.	9-6-11.	9-6-11.
9-11	9-5-4-12.	9-5-4-12.	9-5-4-12.
9-12	10-3-1.	10-3-1.	10-3-1.
10-1	10-8-2.	10-8-2.	10-8-2.
10-2	10-3.	10-3.	10-3.
10-3	10-8-12-4.	10-8-12-4.	10-8-12-4.
10-4	10-8-2-11-5.	10-8-12-4-5.	10-8-12-4-5.
10-5	10-3-1-6.	10-8-2-11-6.	10-3-1-6.
10-6	10-8-12-7.	10-8-12-7.	10-8-12-7.
10-7	10-8.	10-8.	10-8.
10-8	10-3-1-6-9.	10-8-12-4-5-9.	10-3-1-6-9.
10-9	10-8-2-11.	10-8-2-11.	10-8-2-11.
10-11	10-8-12.	10-8-12.	10-8-12.
10-12	11-1.	11-1.	11-1.
11-1	11-2.	11-2.	11-2.
11-2	11-2-3.	11-2-3.	11-2-3.
11-3	11-7-4.	11-7-4.	11-7-4.
11-4	11-5.	11-5.	11-5.
11-5	11-6.	11-6.	11-6.
11-6	11-7.	11-7.	11-7.
11-7	11-2-8.	11-2-8.	11-2-8.
11-8	11-6-9.	11-6-9.	11-6-9.
11-9	11-2-8-10.	11-2-8-10.	11-2-8-10.
11-10	11-7-12.	11-7-12.	11-7-12.
11-12	12-7-11-1.	12-7-11-1.	12-7-11-1.
12-1	12-8-2.	12-8-2.	12-8-2.
12-2	12-8-10-3.	12-8-2-3.	12-8-10-3.
12-3	12-4.	12-4.	12-4.
12-4	12-4-5.	12-4-5.	12-4-5.
12-5	12-7-11-6.	12-7-11-6.	12-7-11-6.
12-6	12-7.	12-7.	12-7.
12-7	12-8.	12-8.	12-8.
12-8	12-4-5-9.	12-4-5-9.	12-4-5-9.
12-9	12-8-10.	12-8-10.	12-8-10.
12-10	12-7-11.	12-7-11.	12-7-11.
12-11			

5.2.2 Experiment 2

This experiment is made to compare the speed and the achievement of optimization heuristics to reach the global optimum point of the 30-nodes network configuration with the demands matrix to be given. The network configuration and the demands matrix for this experiment have been chosen such that the power of flow-based routing to reach a global optimum point against destination-based routing is to be shown. One of the most important aims of this experiment is also showing the power of PGA(W) against GA(W). PGA(W) is also included in this test to show the advantages of parallelizing the fitness evaluation process as told in the third chapter. For PGA(W) test, 7 slave machines are used to paralellize the fitness evaluation process.

The demands matrix for 30-nodes network is chosen such that the optimum point for the routing optimization problem is a case that all links in the given network are fully loaded without any data loss. The cost function used in the simulation is the hybrid cost function mentioned in section 3.4:

$$C_G = a * DLR + \max(\frac{UB(e) - B(e)}{B(e)}) + \frac{\sum_e (\frac{UB(e)}{B(e)})}{|E|}; \quad e \in E : E \rightarrow R^+$$

Each demand has a delay constraint of 24 msec constantly. 30-nodes network is configured such that the solution realizing the optimum point does not comply with shortest path routing constraints, thus destination-based routing cannot reach the optimum point. K-routes and weight system dependent optimization heuristics will be compared in such a case. The demands matrix for 30-nodes network is given in table 12:

Table 12: Experiment 2, demands matrix for 30-nodes network (in units/sec).

	N_0	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_10	N_11	N_12	N_13	N_14	N_15	N_16	N_17	N_18	N_19	N_20	N_21	N_22	N_23	N_24	N_25	N_26	N_27	N_28	N_29
N_0	0	58	0	32	0	0	0	40	0	0	0	0	0	15	0	0	20	0	0	0	0	8	0	0	0	0	0	19	8	0
N_1	0	0	0	70	0	0	25	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	15	0	0	0	0	0	0
N_2	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0
N_3	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	10	0	33	0	5	0	0	20	42	0	0	10	15	13
N_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	28	30	0	0	0	15	0
N_5	0	0	0	0	0	0	0	0	0	5	7	0	0	0	5	23	0	0	0	0	20	0	0	0	0	0	30	0	0	30
N_6	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0
N_7	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0
N_13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	2	0	0	0	0	0	0	0	0	26	0	0	
N_14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	72	0	0	0	0	0	18	0	0	0	10	10	
N_15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
N_16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	10	0	20	30	30	7
N_17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
N_18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	
N_19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
N_20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	35	60
N_21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	0	0	0	10	10	
N_22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	35	0	0	
N_23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	43	
N_24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
N_25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
N_26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
N_27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25
N_28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N_29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Given the 30-nodes network configuration and the demands matrix in table 12, optimization heuristics were applied on the problem. Table 13 shows that all weight system dependent optimization heuristics have not reached close to the optimum point, where K-routes system dependent optimization heuristic GA(K) has reached the optimum point with network cost of 2. Weights adjustment heuristic has been the fastest one among the weight system dependent optimization heuristics. On the other hand, GA(K) has been the most effective one among all, while reaching to the global optimum point. GA(K) has shown that the processing time for optimization of different network sizes does not change drastically as in WA, SAN or GA(W).

Table 13: Performance of optimization heuristics in experiment 2.

	NONE	WA	SAN	GA(W)	PGA(W)	GA(K)
Time(msec)	-	≈1500	≈5700	≈4100	≈2900	≈10000
Cost(units/sec)	122.978	≈86	≈71	≈69	≈69	2

The demand loss rate, maximum, minimum and average utilization values reached at each of the optimization methods are as follows:

Table 14: Example 2, DLR, max. . min. and avg. utilization values for different optimization methods.

	NONE	WA	SAN	GA(W), PGA(W)	GA(K)
DLR	0.12	0.084	0.079	0.068	0
Max. Utilization	1	1	1	1	1
Min. Utilization	0	0.52	0.54	0.52	1
Avg. Utilization	0.76	0.84	0.86	0.86	1

As seen in table 12, not all the nodes have demands destined for others. Thus, the paths only for those nonzero demands are listed in table 15 (those paths that were changed during optimization are highlighted). Since, weight system dependent optimization heuristics could not result in the desired solution, only the GA(K) routing solution is listed together with the startup routing table.

In this experiment we have also seen that PGA(W) has increased the speed of GA(W) with a modification in the fitness evaluation process. PGA(W) has reached the same cost with GA, although it has finished the optimization in three fourth time of GA(W) (4100 to 2900 msec for GA(W) and PGA(W) respectively). On big networks parallelizing proper optimization processes always give better speed performance, however on small networks communication latency between parallel processes may consume more time than a one-node sequential process and the optimization process may even become slower.

Table 15: Experiment 2, routing tables.

S-D	Initial	GA(K)
N_0-N_1	0-1.	0-1.
N_0-N_3	0-2-3.	0-2-3.
N_0-N_7	0-2-3-7.	0-2-3-7.
N_0-N_13	0-2-3-7-13.	0-1-3-4-8-13.
N_0-N_16	0-2-5-11-16.	0-2-5-11-16.
N_0-N_21	0-1-6-12-17-21.	0-1-6-12-17-21.
N_0-N_27	0-1-6-12-17-21-27.	0-1-6-12-17-21-27.
N_0-N_28	0-2-5-11-16-26-29-28.	0-2-5-10-15-20-24-28.
N_1-N_3	1-3.	1-3.
N_1-N_6	1-6.	1-6.
N_1-N_17	1-6-12-17.	1-6-12-17.
N_1-N_23	1-3-7-13-18-23.	1-3-7-13-18-23.
N_2-N_5	2-5.	2-5.
N_2-N_20	2-5-11-16-20.	2-5-10-15-20.
N_3-N_4	3-4.	3-4.
N_3-N_16	3-2-5-11-16.	3-2-5-11-16.
N_3-N_18	3-7-13-18.	3-4-8-13-18.
N_3-N_20	3-4-9-14-15-20.	3-2-5-10-15-20.
N_3-N_23	3-7-13-18-23.	3-7-13-18-23.
N_3-N_24	3-4-9-14-19-24.	3-4-9-14-19-24.
N_3-N_27	3-7-13-17-21-27.	3-7-13-17-21-27.
N_3-N_28	3-7-13-18-23-28.	3-7-13-18-23-28.
N_3-N_29	3-7-13-18-23-28-29.	3-2-5-11-16-26-29.
N_4-N_21	4-8-13-17-21.	4-8-13-17-21.
N_4-N_23	4-9-14-18-23.	4-9-14-18-23.
N_4-N_24	4-9-14-19-24.	4-9-14-19-24.
N_4-N_28	4-9-14-19-24-28.	4-8-13-18-23-28.
N_5-N_10	5-10.	5-10.
N_5-N_11	5-11.	5-11.
N_5-N_15	5-10-15.	5-10-15.
N_5-N_16	5-11-16.	5-10-15-20-16.
N_5-N_20	5-11-16-20.	5-11-16-20.
N_5-N_26	5-11-16-26.	5-11-16-26.
N_5-N_29	5-11-16-26-29.	5-10-15-20-25-29.
N_6-N_12	6-12.	6-12.
N_6-N_17	6-12-17.	6-12-17.
N_7-N_13	7-13.	7-13.
N_10-N_15	10-15.	10-15.
N_11-N_16	11-16.	11-16.
N_12-N_17	12-17.	12-17.
N_13-N_17	13-17.	13-17.
N_13-N_18	13-18.	13-18.
N_13-N_27	13-17-21-27.	13-17-21-27.
N_14-N_15	14-15.	14-15.
N_14-N_18	14-18.	14-18.
N_14-N_24	14-19-24.	14-19-24.
N_14-N_28	14-19-24-28.	14-19-24-28.
N_14-N_29	14-19-24-28-29.	14-15-20-25-29.
N_16-N_20	16-20.	16-20.
N_16-N_24	16-20-24.	16-20-24.
N_16-N_26	16-26.	16-26.
N_16-N_27	16-26-29-28-27.	16-20-24-28-27.
N_16-N_28	16-26-29-28.	16-26-29-28.
N_16-N_29	16-26-29.	16-26-29.
N_18-N_23	18-23.	18-23.
N_20-N_24	20-24.	20-24.
N_20-N_28	20-24-28.	20-24-28.
N_20-N_29	20-25-29.	20-25-29.
N_21-N_23	21-22-23.	21-22-23.
N_21-N_28	21-27-28.	21-22-23-28.
N_21-N_29	21-27-28-29.	21-27-28-29.
N_22-N_23	22-23.	22-23.
N_22-N_26	22-23-28-29-26.	22-21-27-28-29-26.
N_23-N_24	23-28-24.	23-28-24.
N_23-N_28	23-28.	23-28.
N_26-N_29	26-29.	26-29.
N_27-N_29	27-28-29.	27-28-29.

As mentioned before, both destination and flow based routing, thus weight and K-routes system dependent optimization, are capable of reaching the optimum point in experiment 1. On the other side, experiment 2 is made to show that there are some cases where solution space of weight system dependent optimization heuristics is not large enough to cover the global optimum point for the routing optimization problem. First two experiments were made to show the performance of optimization heuristics with static demands matrices. Thus in experiment 3, a simulation is made to show the performance of the optimization heuristics with dynamic demands matrices.

5.2.3 Experiment 3

This experiment is made on the 30-nodes network to compare the adaptation of optimization heuristics to changing demands throughout the network. In other words, this experiment is made to show performance of dynamic routing control together with SAN, WA, weight system and K-routes system dependent genetic optimization heuristics. Network is configured in such a way that some of the nodes in the network create demands to other nodes continuously. The nodes that are effective has the given traffic properties:

Table 16: Experiment 3, node traffic properties.

	Min. demand (units/sec)	Max. demand (units/sec)	Min. period (sec)	Max. period (sec)	Min. inter-arrival (sec)	Max. inter-arrival (sec)	Min. delay constraint (msec)
N_0	3	15	25	75	25	75	24
N_10	2	3	25	75	25	40	24
N_13	1	2	10	10	1	40	24
N_20	3	5	25	50	25	50	24
N_29	3	5	25	35	15	25	24

Each of the effective nodes in table 16, creates demands to all other nodes dynamically according to the following rules:

- Traffic characteristics of each node are determined by 4 values, which are demand magnitude, demand period, demand inter-arrival time and delay constraint.
- Each effective node creates demands to other nodes with a magnitude randomly chosen between minimum and maximum demand size characteristics of that node.
- The demands last for a random period of time in between minimum and maximum period characteristics.

- Between every successive demand for a flow, there exists a random amount of time between minimum and maximum inter-arrival time characteristics.
- Initial routing table is built in such a way that delay constraint shortest paths are used for each of the demands.

This experiment is made on mentioned dynamic network for a simulation time of 100 seconds. The simulation is made along the following 5 situations:

- No optimization: An initial routing table is kept constant along the simulation. Thus a comparison between non-optimized and the optimized network.
- GA(K): K-routes system dependent genetic algorithm is used to optimize the network continuously.
- GA(W): Weight system dependent genetic algorithm is used to optimize the network continuously.
- SAN: Simulated annealing meta-heuristic is used to optimize the network continuously.
- WA: Weights adjustment heuristic is used to optimize the network continuously.

At each second, cost of the network is sampled for each of the optimization heuristics. The cost function used in the simulation is the hybrid cost function mentioned in section 3.4:

$$C_G = a * DLR + \max\left(\frac{UB(e) - B(e)}{B(e)}\right) + \frac{\sum_e \left(\frac{UB(e)}{B(e)}\right)}{|E|}; \quad e \in E : E \to R^+$$

We can compare the performances of the heuristics together with the non-optimized case. Figure 11 and 12 give the total demand and number of demands versus time information respectively. Figure 13, 14, 15 and 16 give the simulation results according to network cost, demand loss rate, maximum and average utilization respectively. Given graphs can help us compare the optimization heuristics. In figures 13,14,15 and 16, it can be seen that non-optimized network performance values are given as a reference to the heuristic performances. Tables 17, 18 and 19 shows statistical values regarding DLR, number of optimizations made and average run-time values respectively.

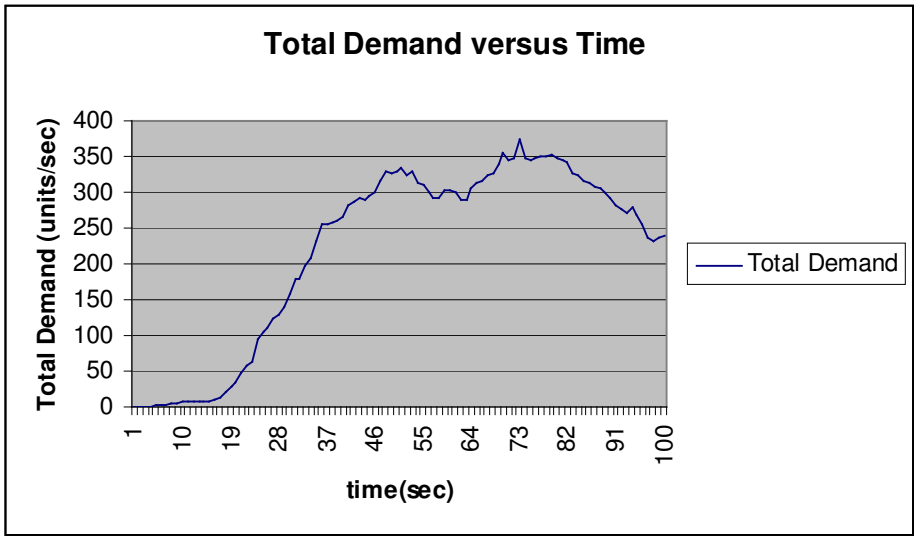


Figure 11: Experiment 3, total demand versus time graph.

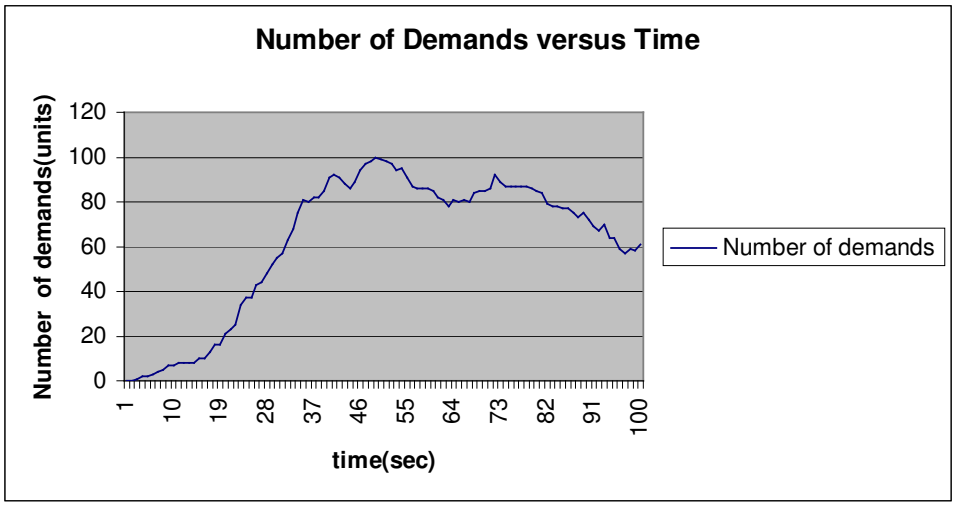


Figure 12: Experiment 3, number of demands versus time graph.

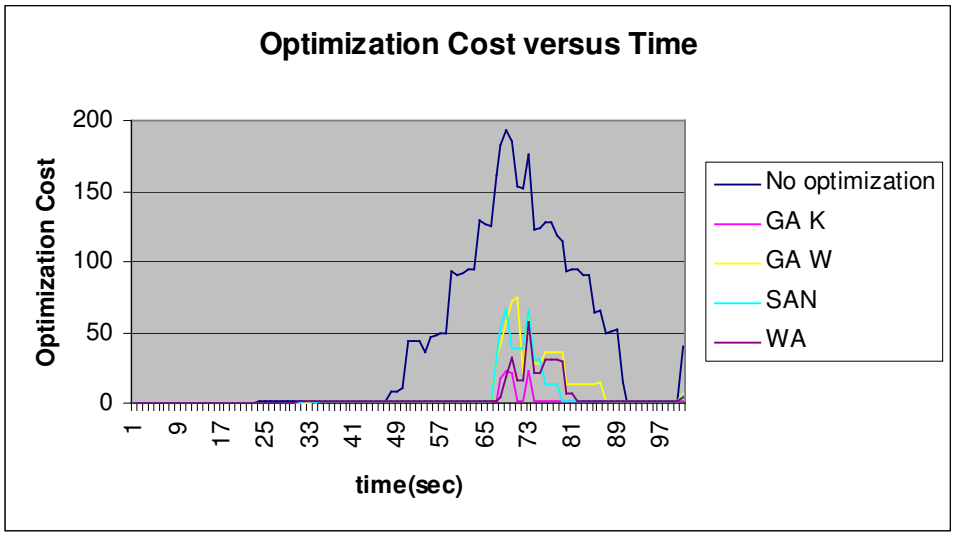


Figure 13: Experiment 3, optimization cost versus time graph.

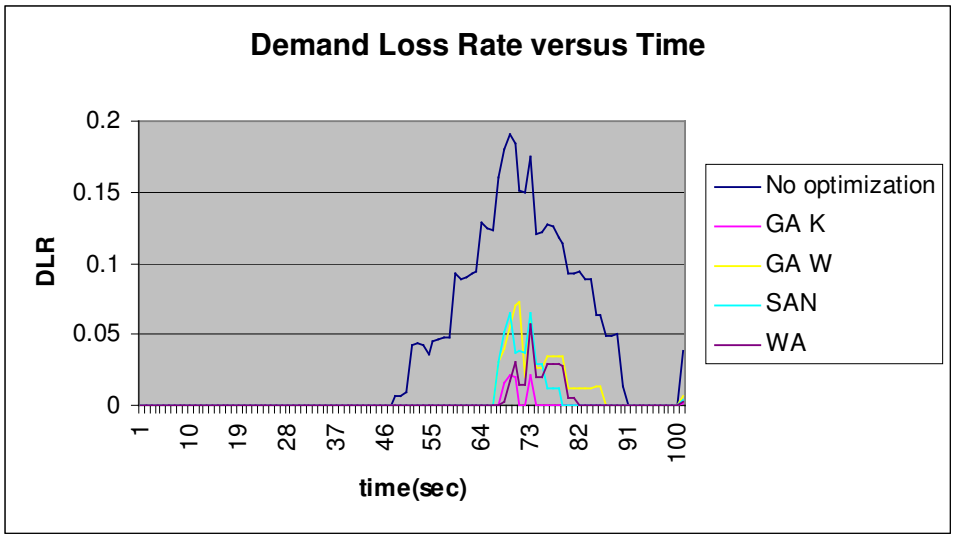


Figure 14: Experiment 3, DLR versus time graph.

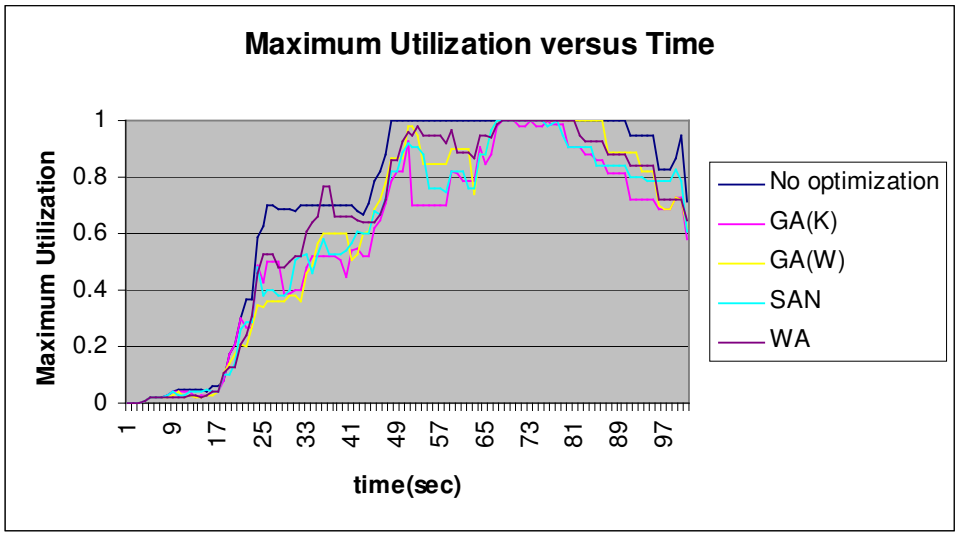


Figure 15: Experiment 3, maximum utilization versus time graph.

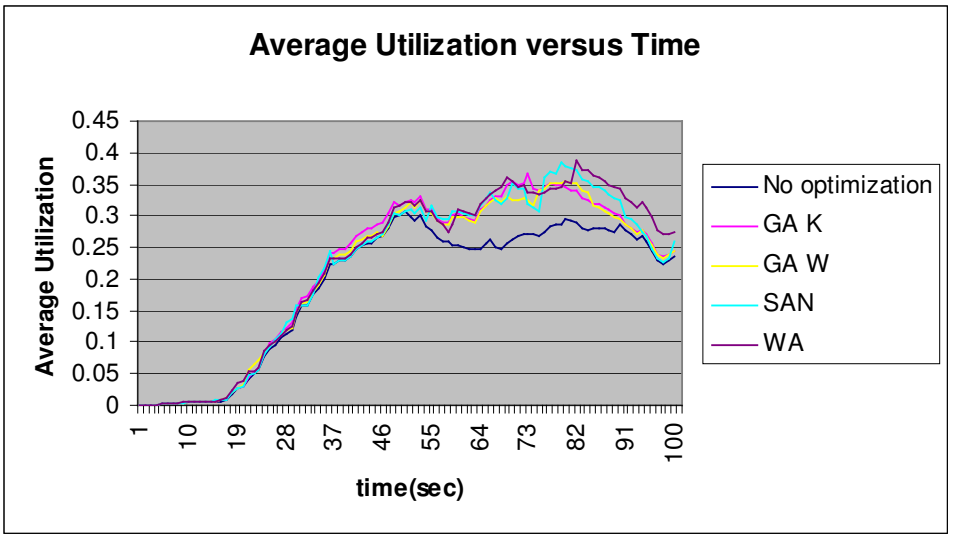


Figure 16: Experiment 3, average utilization versus time graph.

Table 17: Experiment 3, average DLR for different heuristics.

	No optimization	GA(K)	GA(W)	SAN	WA
Avg. DLR	0.038673	0.000771	0.006256	0.004161	0.003035

Table 18: Experiment 3, number of optimizations made, accepted and denied on the network.

	No optimization	GA(K)	GA(W)	SAN	WA
Optimizations made	0	27	23	33	50
Optimizations accepted	0	21	17	25	45
Optimizations denied	0	6	6	8	5

Table 19: Experiment 3, average runtime of algorithms.

	No optimization	GA(K)	GA(W)	SAN	WA
Avg. runtime (msec)	0	2060	2599	1008	228

According to the graphs given in figures 13 and 14, K-routes system dependent genetic algorithm has given the best results according to optimization cost, demand loss rate and maximum link utilization values.

As seen in figures 13, 14 and table 17, WA heuristic has been the best for decreasing demand loss rate in weight system dependent methods. On the other hand, WA heuristic has not been successful in decreasing maximum link utilization as seen in figure 15. Table 18 shows the number of optimizations made online on the network during 100 seconds. As seen in tables 18 and 19, it is obvious that WA heuristic has been the fastest one among all optimization methods. However we see that there is no direct relation with the performance of a heuristic and the number of times it has run. This situation is directly related to the WA structure, which is suitable for preventing overloading but does not have an intelligent method for decreasing maximum link utilization.

Among weight system dependent optimization methods, SAN meta-heuristic has been best in minimizing maximum link utilization value. On the other hand, GA(W) has been the worst among the optimization methods. In fact, GA(W) has been slow for optimizing the network cost and given fewer number of successful updates than the other heuristics (see tables 18

and 19). It has been observed that GA(W) had a decrease in the speed when the number of flows has reached the maximum value at time 50 (see figure 12).

To sum up, GA(K) has shown better performance than GA(W), WA and SAN during online optimization simulation made in this experiment. GA(K) has obviously decreased the average DLR of the network more than GA(W), WA and SAN. Success of GA(K) in preventing data loss is because of the routing freedom of flow-based routing. But, these results would surely change in other network configurations, with more nodes and different demands characteristics.

The aim of dynamical routing control is not only preventing data loss on the network but also minimizing the maximum link utilization to make space for new demands. Applying such a load balancing and working on best utilization of the network lets the network to service for unexpected traffic variations in between dynamical routing control updates. If we inspect figures 14 and 15 together we can divide the simulation timescale into two portions:

- No-overloading period: This period is from time 0 to time 49 and time 92 to time 100.
- Overloading period: This period is from time 50 to time 91.

We have seen that in the overloading period, all the optimization methods have decreased the overloading to an acceptable value. On the other hand, in the no-overloading period, we see that optimization methods also decreased the maximum link utilization at most %30 (referencing GA(K) at time 29).

Thus, dynamical routing control is worth to use for best utilization of the network resources at hand. The optimization heuristic to be applied on a specific network should be chosen according to the heuristic's speed, its performance on the given network, dynamic routing control refresh rate, and the ability of the network to run destination or flow based routing methods. Simulations will certainly be helpful for selecting dynamic routing control parameters for a given network.

CHAPTER 6

CONCLUSION

In this study, routing optimization methods for communication networks have been discussed. Thus, different optimization methods for preventing data loss on overloaded communication links and methods for utilizing the network resources efficiently have been introduced and routing optimization problem has been defined.

This study has started with making an introduction on the traffic characteristics of current communication networks and the need for network routing optimization. In the second chapter, an introduction for dynamical routing control systems has been made and the reasons for using such management methods have been discussed. In the third chapter, network routing optimization problem has been defined and a discussion of the mathematical problem has been made. In the fourth chapter, optimization heuristics for solving the routing optimization problem has been explained by making an analogy between dynamic routing control and the optimization methods. The experimental work and performance analysis of the optimization heuristics has been made finally in the fifth chapter. Every step for developing a dynamical routing control system has been stated in order from motivation to implementation and analysis phases throughout the study.

As mentioned before, dynamic routing control is used to utilize the network efficiently and it includes methods for serving routing solutions conforming to different kinds of routing methods. In this context, a simulation tool, DRCSP, including all proposed optimization heuristics has been designed and developed as the implementation of this study. This implementation includes an event driven simulation tool with five different heuristic implementations developed in the context of weight and K-routes system dependent routing optimization methods.

Optimization heuristics and their performance analysis has been the main objective during this study. In this study delay constrained cost optimization routing problem of path-constrained path-optimization class is chosen. The objective function of optimization methods has been formed such that it includes demand loss rate and utilization related parameters. Thus the aim of the optimization methods has been preventing data loss and

minimizing maximum link utilization of the network to make space for new demands. In this context, optimization methods for two different routing strategies have been implemented and developed as a solution tool for the routing optimization problem. It has been shown that both weight and K-routes system dependent routing optimization heuristics are conforming to destination and flow based routing methods respectively:

- Destination-based routing:
  - WA: Weights adjustment heuristic,
  - SAN: Simulated annealing meta-heuristic,
  - GA(W) and PGA(W): Genetic algorithm (weight system) and parallel GA.
- Flow based routing:
  - GA(K): Genetic algorithm (K-routes genetic system).

Destination and flow based routing methods have been explained in details and both advantages and disadvantages of each have been discussed in the second chapter. In fourth chapter, WA, SAN, GA(W), PGA(W) and GA(K) have been explained in details and finally an analysis has been made in the fifth chapter. DRCSP simulation results for the heuristics on different network configurations have shown the typical characteristics of destination and flow based routing methods, in other words, weight and K-routes system dependent optimization methods.

Analysis results have shown that, optimization heuristics working on flow-based routing method have resulted in better network performance than that of heuristics working on destination-based routing method. We can classify the performance of the implemented optimization methods as follows:

- Minimizing demand loss rate: GA(K) has shown the best performance in minimizing demand loss rate of a dynamic network as mentioned in the third example. On the other hand, weight system dependent optimization methods also decreased the demand loss rate but could not reach the performance of the K-routes system dependent genetic algorithm's performance.
- Success in load balancing: GA(K) has also shown the best performance in load balancing and has been successful in minimizing the maximum link utilization as mentioned in the third example. Although weight system dependent optimization methods were successful in minimizing the maximum link utilization during no-overloading period, they have not reached the performance of the K-routes system dependent genetic algorithm during the simulation. As told in the third example,

weights adjustment heuristic could not even decrease the maximum link utilization because of the insufficient capabilities of the algorithm.

- Speed performance: Weights adjustment heuristic has been the fastest one among implemented optimization methods. However, this heuristic has not shown the same performance in load balancing area although it has been successful in decreasing demand loss rate to some degree. One of the interesting sides of this study was the implementation of a parallel genetic algorithm for the routing optimization problem. PGA was designed and implemented such that GA gained the ability of distributing overwhelming fitness evaluation process to slave machines and an increase in the speed of the algorithm was observed. Thus, optimizing large networks shall become easier through PGA.

To sum up, K-routes system dependent genetic algorithm working on flow-based routing method has shown the best results in the global optimum point convergence of the routing optimization problem. Thus, optimization heuristics working on flow-based routing method are likely to be used widely when flow based routing methods will be dominant in communication networks. However, weight-system dependent optimization methods are still in consideration because of their sufficient performance and robust characteristics on current networks running destination-based routing protocols. One of the most important points on selecting optimization methods for a dynamic routing control should be the general traffic characteristic of the communication network and its routing strategy. On the other hand, for better management of networks, dynamic routing control should also be configured appropriately, in terms of probing network information, re-routing strategy and control refresh rate.

Developing robust, fast and efficient heuristics for routing optimization problem is a must in today's communication networks. Communication networks are growing fast and dynamic nature of traffic demands on these networks requires that dynamical routing control be made faster. Thus, speed and performance requirements for dynamic routing control are increasing day by day. Speed constraint of dynamic routing control requires that the optimization should finish as soon as possible to make the system both state dependent and adaptive. In order to increase the speed of the proposed algorithms, different code optimizations, distributed programming techniques should be used. Thus, future work for this study includes design of parallel optimization heuristics through distributed programming techniques.

REFERENCES

[1] N. Shimamoto, A. Hiramatsu, K. Yamasaki, “A Dynamic Routing Control Based on a Genetic Algorithm”, IEEE, 1993.

[2] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, "Requirements for Traffic Engineering over MPLS", RFC 2702, September 1999.

[3] D. Awduche, "MPLS and Traffic Engineering in IP Networks", IEEE Communications Magazine, Dec. 1999.

[4] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, X. Xiao, “Overview and Principles of Internet Traffic Engineering”, RFC 3272, May. 2002.

[5] Eric Crawley, Raj Nair, Bala Rajagopalan, Hal Sandick, “A Framework for QoS-based Routing in the Internet”, draft-ietf-qosr-framework-00.txt,March, 21, 1996.

[6] D. Mitra and K.G. Ramakrishnan, "A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks", Proc. Globecom'99, Dec 1999.

[7] P. Cortes, J. Munuzuri, J. Larraneta, L. Onieva, “A Genetic Algorithm Based on Cell Loss for Dynamic Routing in ATM Networks”, Seville University, 1999.

[8] Anton Riedl, “A Hybrid Genetic Algorithm for Routing Optimization in IP Networks Utilizing Bandwidth and Delay Metrics, Institute of Communication Networks, Munich University of Technology, 2003.

[9] Anton Riedl, “Routing Optimization and capacity Assignment in Multi-Service IP Networks”, Institute of Communication Networks, Munich University of Technology, 2003.

[10] Henrik Abrahamsson, Bengt Ahlgren, Juan Alonso, Anders Andersson, and Per Kreuger, “A Multi Path Routing Algorithm for IP Networks Based on Flow Optimization”, Swedish Institute of Computer Science, SICS.

[11] M. Pioro, A. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, S. Kozdrowski, “On open shortest path first related network optimization problems”, Elsevier Science, 2002.

[12] T. A. Al-Qahtani, M. J. Abedin, S. I. Ahson, “Dynamic Routing in Homogenous ATM Networks Using Genetic Algorithms, IEEE, 1998,

[13] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, M. Thorup, “A Hybrid Genetic Algorithm for The Weight Setting Problem in OSPF/IS-IS Routing”, AT&T Labs, June 2003.

[14] E. Mulyana, U. Killat, “An Alternative Genetic Algorithm to Optimize OSPF Weights”, DCN, Technical University Hamburg, 2002.

[15] H. Kanoh, T. Nakamura, “Knowledge Based Genetic Algorithm For Dynamic Route Selection”, Institute of Information Sciences and Electronics, University of Tsukuba, IEEE, 2000.

[16] C. Devies, P. Lingras, “Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks”, Dep. Of Mathematics and Computing Science, Saint Mary’s University, Elsevier Science, 2002.

[17] A. T. Haghigkat, K. Faez, M. Dehghan, A. Mowlaei, Y. Ghahremani, “GA-Based Heuristic Algorithms for QoS Based Multicast Routing, Elsevier, 2003.

[18] D. Staehle, S. Köhler, U. Kohlhaas, “Towards an optimization of the routing parameters for IP networks”, University of Wuerzburg, Report No: 258 May. 2000.

[19] L. T. M. Berry, S. Köhler, D. Staehle, P. Tran-Gia, “Fast heuristics for optimal routing in large IP networks”, University of Wuerzburg, Report No: 262, July 2000.

[20] M. Ericsson, M. G. C. Resende, P. M. Pardalos, “A Genetic Algorithm For The Weight Setting Problem In OSPF Routing”, October 9 2001.

[21] W. Ben-Ameur, N. Michel, B. Liau, “Routing Strategies for IP Networks”, France Telecom R&D, 2001.

[22] B. Fortz, M. Thorup, “Internet Traffic Engineering by Optimizing OSPF Weights”, IEEE INFOCOM, March 2000.

[23] Shigang Chen, Klara Nahrstedt, “An Overview of Quality of Service Routing for Next-Generation High Speed Networks: Problems and Solutions”, University of Illinois at Urbana Champaign, IEEE Network, November/December 1998

APPPENDIX A

DYNAMIC ROUTING CONTROL SIMULATION PROGRAM: DRCSP

A.1 Introduction

In this chapter, the Dynamic Routing Control Simulation Program, abbreviated as DRCSP, which is the practical work done in accordance with the theoretical part of the whole work, is to be told. The DRCSP software will be explained in details and user graphical interface is told also.

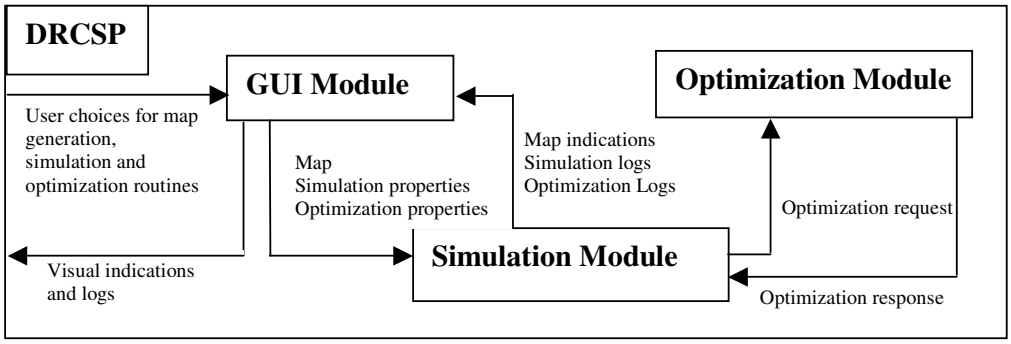
DRCSP is designed in such a way that user can create network topologies, give traffic properties to the network and simulate the transmission on the network while he can use different optimization methods, to control the network. DRCSP consists of three main modules: graphical user interface, simulation and optimization modules. Network maps, which are created in map generator GUI and saved in .map file format, can be used to start a simulation in simulator module. As simulation is executed, optimization module works to decrease network cost in parallel and simulator module logs every event and network cost characteristics in .txt format respectively. Thus, each simulation can be criticized by the help of the log files. In the next section the software structure of the DRCSP is explained.

A.2 Software Structure of DRCSP

DRCSP software consists of three main threads, main thread, simulation thread and optimization thread. These three threads include the three main modules GUI, simulation and optimization modules respectively. GUI is responsible for management works such as network map generation, simulation and optimization properties selection, and user information services such as logging and graphical indications. Simulation module is the main worker thread which generates traffic on the given network map file according to the simulation properties, calls the optimization module functions as needed, and passes the simulation instance information to the GUI module. Finally the optimization module, which can also be called as the slave worker thread for the simulation module, accepts network optimization requests, executes pre-selected optimization methods and finally passes the

optimization solution to the simulation module while logging every step. You can see the software structure in the table given below.

Table 20: DRCSP software modules.



All three modules will be explained in the following sub-sections.

A.2.1 GUI Module

GUI module is responsible for management works such as network map generation, simulation and optimization properties selection, and user information services such as logging and graphical indications. GUI module consists of two forms, main form and the map form.

A.2.1.1 Main form

Main form is the place where all simulation and optimization accesses can be made. Below is the main form figure:

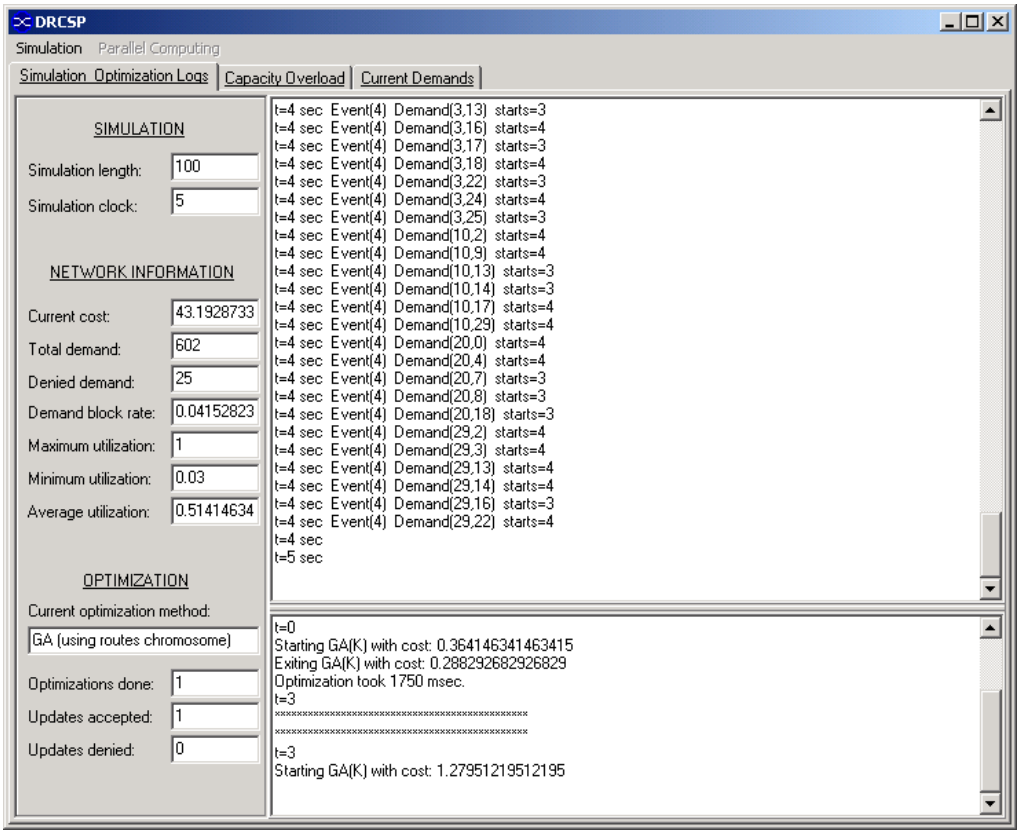


Figure 17: Main form of DRCSP.

In the main form user can:

- Open a network map file for simulation,
- Adjust simulation parameters,
- Decide on dynamic routing control to be used and adjust method parameters.
- Start a simulation, stop, pause or resume it,
- Open the map form to create a network map file,
- Adjust and start parallel processing services,
- Whenever a simulation is started user can access:
    - Simulation and optimization standard outputs or logs in the simulation/optimization logs page.
    - Capacity usage and capacity amount information of every link of the network on the capacity usage/capacity page,
    - Current inter-nodes demands matrix in the current demands page.

- Be informed about simulation statistics, such as simulation length, simulation clock, current cost of the network and number of optimizations made, accepted or denied.

A.2.1.2 Map form

Map form is the place where you can see the simulated network map, create a new map, open an existing one, change and save it. During a simulation user can follow the link overloading status from the colors of the links shown on the map.

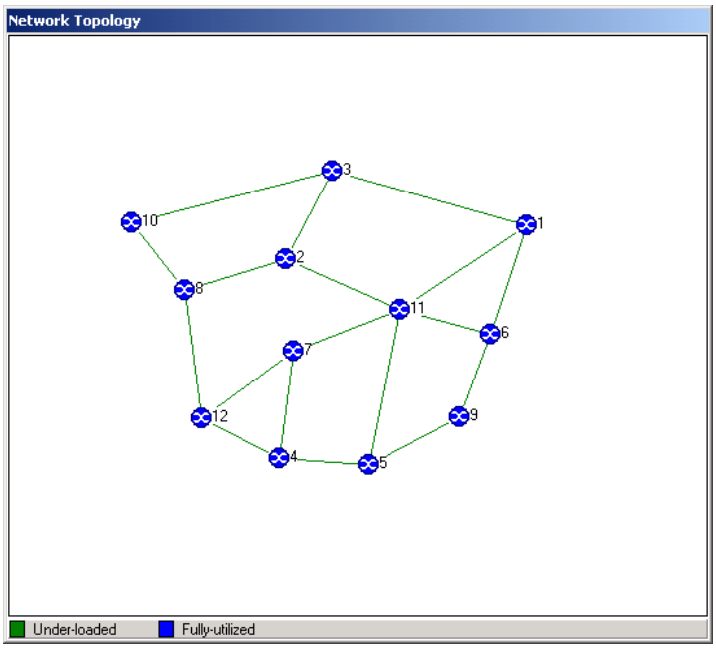


Figure 18: Map form of DRCSP.

Usage of the GUI will be discussed thoroughly, map creation, simulation and optimization processes will be told step by step later in this chapter.

A.2.2 Simulation Module

Simulation module is the main worker thread, which is responsible for generating traffic on a given network map, calling optimization routines whenever needed, logging every event in the simulation, and passing information to the GUI module. Simulation module is, on both ends, connected to the GUI and the optimization modules. GUI interface connection is used to transfer network map, simulation properties, optimization properties and simulation log messages. Optimization module connection is used to pass current network information to the optimization routine including connection matrix, routing table and current demands table to initialize the optimization process and wait for the updated connection and routing tables from the optimization module. This interface is also used for passing optimization log messages to the GUI.

Traffic generation capability of the simulation module is strongly related to the network map passed by the GUI. DRCSP uses the content of the map file to simulate the network and generates traffic while evaluating the network cost and calling the optimization routines. It is better to start with the relation between the map file and the simulation module. The information read from the map file is as below:

- Number of nodes on the network,
- Properties of each node that are located on the network map:
    - Node's name,
    - Node's coordinates,
    - Node's traffic generation properties (meaningful if node is effective) including:
        - Demand generation property,
        - Demand's arrival rate properties,
        - Demand's delay constraint properties,
        - Duration properties for each demand
- Connection matrix defining the links between the located nodes where each links' properties such as reliability, weight, delay, and bandwidth are registered.
- Initial demands matrix, containing information of all inter-nodes initial demands.

Simulation thread is initialized with the connection matrix and the node properties first. At the very beginning, initial routing table is formed, and all nodes are assumed to be silent for a while. Thus, event based-simulation always starts with the initial inter-nodes demands and as

time passes, each of the nodes begins to generate traffic. This design specific topic is discussed later.

There are two actions performed in the simulation module of the program. The first action of the module is the traffic generation, where process continues to work and generate traffic until a stop interrupt. On the other side, there is another action, where the simulation process checks if the network is overloaded and calls the optimization process from the optimization module, which tries to optimize the network as rapidly as possible. This architecture is explained in the next section.

Using the map information simulator starts a traffic event generation task called as simulation process. As simulation process executes, it checks for the overall network cost for the possibility of an overloaded network. Overloading any link in the topology, exceeding its bandwidth, results in an overloaded network. If an overloaded network exists, the demands cannot be routed properly, such that some of them are discarded at the routing nodes that have overloaded links on the way. Thus simulation process calls the optimization process and goes on generation process until the optimization process sends an update request. Let's discuss the simulation process extensively in the next subsection.

A.2.2.1 Simulation Process

Simulation module is responsible for initializing a simulation process and run it with given parameters. User-selected map file is first decoded to form the nodes array, connection matrix, initial inter-nodes demands array and the routing table. Nodes array includes each node's traffic generation properties, which will be used create events in the simulation process. Events are consumed or created while simulation process at each instance evaluates network cost and probe the network according to the given connection matrix, routing table and the demands matrix.

To understand the simulation process, the event-driven structure of the program must be told in details. In the next three sub-sections, first the definition of an event is made, secondly the structure of the linked list holding the events is discussed, and finally how the simulation works on the given linked list is explained.

A.2.2.1.1 Events

Simulation process, or the simulation thread is the main worker thread, which runs with pre-created events, continuously consumes old events or creates new events. All events created in the initialization period of the simulation process are listed in a linked list according to the event finish times. Let us first examine the structure of an event. An event includes 6 basic properties, which are held in the structure:

- Event Type
- Finish Time
- Bandwidth
- From
- To
- Next Event

A.2.2.1.1.1 Event Type

Event type is the main property of the event structure that the simulation process examines first to take an action. There are several event types defined in the simulation thread context. These are:

- Stream: When this type of event is created, a data transfer of event's bandwidth value is started on the network. There are two kinds of streams according to the bandwidth value:
  - Zero Stream: Zero streams can be defined as the silent period of data communication between an initiator node and a destination node.
  - Non-zero Stream: Non-zero streams are the data transfer events between an initiator node and a destination node.

  According to the simulator's architecture concept, every effective node has a continuous stream in either zero or non-zero state. All along the simulator timeline, a node's stream process is never broken but stream events of "zero" – "non-zero" – "zero" – "non-zero" goes on and on. At the very beginning of the simulation, just before the simulator's clock is started, every node is checked if it is effective and if so, initial zero stream events are created for effective nodes for every destination node. So, at the beginning, every effective node has a silent period of random length to destination nodes. These silent periods will last in the timeline and they will be replaced by non-zero stream events, like actual data transfers.

- Control**:** This event type is reserved for future implementations, has no effect now.
- Update: This event type can only be created if the optimization thread has an updated network with a decreased network cost. At every tick of the simulator timeline, simulator thread checks if optimization thread has finished processing (if it has an update) and if so, simulator thread schedules an update event for the next time tick. This process (scheduling update to a later time: 1 tick now) simulates the routing information updating delay in the actual case.
- Link broken: This is for future use. This event type can only be scheduled if a link between two nodes is prone to link errors, and can be broken. Recall that every link has a reliability property in between 1 and 5, where 1 is the most reliable case. This reliability property can be used for scheduling link broken events for every non-reliable link (links having reliability 2, 3, 4,5). This event is currently not applicable.
- Link recovered: This is for future use. This event type simulates the recovery of a broken link. When a link_broken event is consumed, a link_recovered event is scheduled to a later time. This event is currently not applicable.

(Note: At the very beginning of the simulation, just before the simulator clock is started, link_broken events are scheduled for every non-reliable link to be processed at a later random time. Every link_broken event creates a link_recovered event and vice versa.

A.2.2.1.1.2 Finish Time

This property shows the finish time of a created event. This property can be examined for event types separately:

- Finish time for stream event: Finish time states that the stream will end at that time tick.
- Finish time for control event: A control event is a "1 tick period event" and has no duration. This event is scheduled to be started, processed and finished at that time tick. Control event is scheduled just before the simulator's clock started, to a finish time of "0+ CONTROL_ARRIVAL_THRESHOLD" and every control event triggers a control event scheduling to a finish time of "NOW+CONTROL_ARRIVAL_THRESHOLD" and this goes on and on.
- Finish time for update event: An update event is scheduled to be processed at finish time. This event is also a "1 tick period event". The crucial point in network updating is that the processing delay is simulated as scheduling the update event to "NOW+delay".

- Finish time for link-broken event: This is for future use. Link broken event is a 1 tick period event that is consumed at finish time. It surely schedules a link _recovered event to a later random time considering link's reliability.
- Finish time for link-recovered event: This is for future use. Link recovered event is a 1 tick period event too. It schedules a link_broken event to a later random time considering link's reliability.

A.2.2.1.1.3 Bandwidth

This property is only used for stream events, and is filled –1 for every other event. Bandwidth for a stream is 0 for a zero stream and a positive number for a non-zero-stream.

A.2.2.1.1.4 From

This property is meaningful for stream, link_broken and link_recovered events. It is filled with –1 for every other event type. It shows the source for the stream event, and link's one end for the link_broken and link_recovered events.

A.2.2.1.1.5 To

This property is meaningful for stream, link_broken and link_recovered events. It is filled with –1 for every other event type. It shows the destination for the stream event, and link's one end for the link_broken and link_recovered events.

A.2.2.1.1.6 Next Event

This property is used for linked list linking. It shows the next event in the link list. Linked list consists of events that are sorted according to their finish times and the first event is the head of the linked list. Consumption of every first event causes the first event to be shifted one, to its next event.

A.2.2.1.2 Linked List

Simulation process is designed as an event driven process, which schedules events in a linked list and executes them in the time-line. As described before, each event has a finish time property, and this property is used to push the event in the linked list. Linked list is arranged in increasing finish time properties. At each tick of the clock, simulation process pops the first event in the linked list if the finish time of the event is equal to the clock value. Let's describe the structure of the linked list with the table given below.

Table 21: Linked list of events.

Event Name	Event Type	Finish Time	Bandwidth	From	To	Next
eventA	control	1000	-1	-1	-1	eventB
eventB	stream	1912	9	4	8	eventC
eventC	stream	2238	22	3	5	eventD
eventD	stream	2245	0	5	9	eventE
eventE	stream	2278	21	2	14	eventF
eventF	stream	2543	5	8	3	eventG
eventG	stream	3002	8	12	7	eventH
eventH	link_broken	3312	-1	5	7	NULL

Above table is a linked list where each row addresses the next row as the next event to be handled. All events are of type stream, where some of them are of demand "0" and some of them are of demand bigger than "0". "0" demand means that the stream between From and To nodes will remain silent for some time (till finish time). Non-zero demand means that there is a demand flow consuming link bandwidths on the way. Actually, zero demands simulate the inter-arrival time (randomly chosen from the traffic arrival distribution function) between consecutive demands between two same nodes, and the non-zero demands simulate the link consumption (for a while randomly chosen from the traffic period distribution function).

A.2.2.1.3 How Simulation Process Works

In this section, how the simulation process works is discussed. First of all, simulation process basically does two things:

- Consuming events,
- Checking network if links are overloaded and calling optimization routines.

These two actions are called continuously at every tick of the simulation clock. In the next two subsections, first the consumption of events in the event linked list and then the optimization routine call actions are discussed. Table 22 includes the pseudo-code for the simulation process:

Table 22: Simulation process.

```
1.    SimulationProcess()
2.    {
3.      local_clock=0;
4.      while(!Terminated) /*management layer can terminate this routine,
5.                                    sets simulation thread->terminated=true*/
6.      {
7.        if(FirstEvent->finish = = local_clock) /*check if there is a scheduled event
8.                                                                          for now*/
9.        {
10.         if(FirstEvent->type = = stream) /*There is a stream event in queue*/
11.         {
12.           if(FirstEvent->BW >0) /*the end of an interarrival period*/
13.           {
14.             remove killed demand from all the links on the route
15.             replace event with a non-zero demand event finishing at random time
16.             update demands matrix
17.           }
18.           else if(FirstEvent->BW >0) /*end of a demand period*/
19.           {
20.             remove killed demand from all the links on the route
21.             replace event with a zero demand event finishing at random time
22.             update demands matrix
23.           }
24.         }
25.         else if(FirstEvent->type = = update)
26.         {
27.           update all connection matrices
28.           evaluate new cost according to new routing table
29.         }
30.         else if(FirstEvent->type = = control)
31.         {
32.           make necessary controls
33.         }
34.       }
35.      check_if_update_needed(); /*check if optimization thread finished working
36.                                    if so schedule an update event for the next tick
37.                                    if not check if  network is overloaded
38.                                    if network is overloaded start an optimization process*/
39.       if(FirstEvent->finish!=local_clock )
40.       {
41.         local_clock++;
42.       }
43.     }
44.   }
```

A.2.2.1.3.1 Consuming Events

Simulation process runs a local clock, where at each tick the event list's first event is checked if that event will be consumed at that time. When a stream event with zero demand is consumed, a stream event with non-zero demand is scheduled for a randomly chosen finish time. In the same way, when a stream event with non-zero demand is consumed, a stream event with zero demand is scheduled for a randomly chosen finish time. All stream events begin at the time they are created, consume link capacities etc., and end at the time written in their finish time property.

Assume that first event in the linked list is a stream from 1 to 10 with demand 23 ending at finish time 123. Also assume that current simulation thread time is 122, and thread is about to tick to time 123. When simulation thread time equals 123, it is checked if the first event in the linked list ends at current tick. In this example stream from 1 to 10 with demand 23 ends at finish time 123. Thus, simulation thread ends this non-zero flow and replaces it with a silent flow (simulating inter-arrival time) of zero demand event. So the new demand is, say, stream from 1 to 10 with demand 0 ending at finish time 1982. When simulation thread current time ticks to 1982 this zero demand event will be replaced with a non-zero demand event, and the simulation will go on like this. So:

- Inter-arrival times are simulated by zero demand stream events,
- Flow durations are simulated by non-zero demand stream events.

A.2.2.1.3.2 Calling Optimization Routines

At every tick of the simulation, simulation process checks if the network is overloaded. If it decides that the network is overloaded, it checks for any optimization thread started before. If there is not any optimization process running, simulation process calls for an optimization process to reduce the network cost. As a rule of thumb, at any time, only one optimization thread can execute. While simulation thread continues executing, the optimization thread can last; thus simulation thread must be informed that the optimization thread offers a new network routing table for a possible less-cost network. This type of event is the **update** event. Any type of event can be followed by another (update-stream-stream-control-link_broken-control-control-etc.). An update event is as below:

Table 23: An update event.

Event Name	Event Type	Finish Time	Bandwidth	From	To	Next
EventX	update	1253	0	0	0	eventY

When simulation thread consumes an update event, it updates all connection matrices, routing tables with the optimized ones. An update event can only be created by an optimization thread. Thus when update is consumed, there is not any possibility of creating another update event replacing it.

A.2.3 Optimization Module

The crucial point in DRCSP architecture is the usage of built in optimization routines to optimize the routing table such that the network cost is minimized. If simulation process decides that the network is overloaded, it starts optimization process to minimize the network cost. Optimization process continues execution in parallel with the simulation process. When the optimization process finishes execution it releases a flag to inform the simulation process to update the routing table in the next iteration.

Optimization process possibly changes the paths the demands are routed in such a way that the network cost is minimized. The key concept is routing the demands on the less-loaded paths. Optimization process needs some time to execute and find a hopefully minimized cost. This time depends mainly on the number of nodes and links on the topology. This processing time causes the simulation process to execute with the overloaded network for some time and some demands are absolutely lost during this time. Thus, in such applications main aim is the minimization of the optimization thread processing time to avoid losing data on the network. Let's now look at the optimization process implementation in table 24:

Table 24: Optimization process.

```
1.  OptimizationProcess()
2.  {
3.    Initialize optimization matrices with the matrices given by simulation thread;
4.    Start the selected optimization method
5.    Update the matrices with the optimized ones.
6.    Release the update flag so that simulation thread schedules an update event.
7.  }
```

At every call of the optimization process, previously selected optimization method is executed and the results are passed to temporary matrices, which will be used by the simulation process. At the end of the simulation process, a flag is released to inform the simulation process that the optimization process has just finished execution. Optimization process properties are adjustable through the graphical user interface and can be accessed during the simulation.

In the GUI, user can adjust optimization module parameters and start the simulation according to his needs. The implementation can be seen in figure 19:

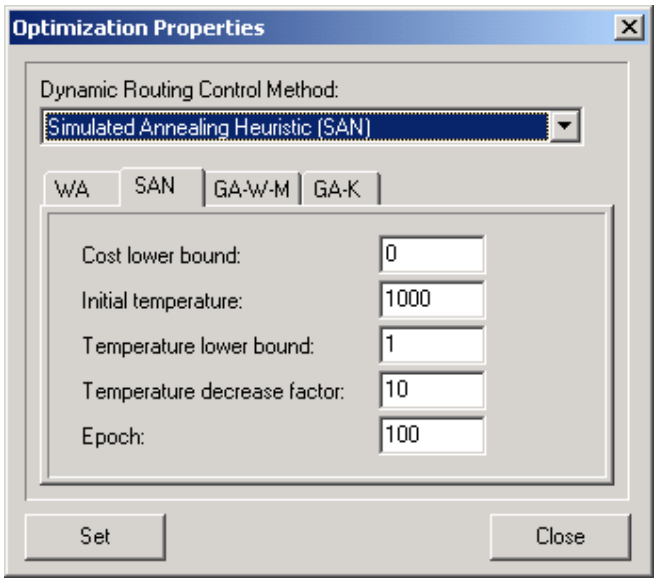


Figure 19: Adjusting optimization properties.

The parameters that can be adjusted by the user are as follows:

- Optimization method: Optimization methods that can be used in DRCSP are consistent with the heuristics discussed in the previous chapter. However, the pseudo-codes or state flow charts given for the heuristics may be slightly different in the implementation. Those heuristics that were proposed in other works, [1], [8], [9], [11], [12], [13], [14], [15], [20], have been inspected and experimented for constructing the optimization heuristics implemented in this study. Some modifications have also been made on the heuristics to increase speed and to give better results. The optimization heuristics implemented are:
  - Weights Adjustment Heuristic (WA)
  - Simulated Annealing Heuristic (SAN)
  - Genetic Algorithm (using weights chromosome)
  - Parallel Genetic Algorithm (using weights chromosome)
  - Genetic Algorithm (using routes chromosome)
- Parameters for optimization heuristics: Each of the optimization methods can be configured with the parameters available for that method:
  - Weight adjustment: The only configurable parameter for this method in DRCSP is the epoch size.
  - Simulated Annealing: Cost lower bound, initial temperature, temperature lower bound, temperature decrease factor and epoch size are the configurable parameters of this heuristic in DRCSP.
  - Genetic Algorithm Modified (using weights chromosome): Population size and the maximum number of generations are the configurable parameters of this method in DRCSP.
  - Parallel Genetic Algorithm Modified (using weights chromosome): This method uses the same configuration parameters with Genetic Algorithm Modified.
  - Genetic Algorithm Modified (using routes chromosome): Population size, maximum number of generations and maximum number of routes, K, are the configurable parameters of this method in DRCSP.

Optimization methods used in DRCSP may give different results with different network configurations and traffic loads. As mentioned before, weight system dependent mechanisms and K-routes system dependent mechanisms use destination-based and flow-based routing

methods respectively. This situation, itself, may cause the optimizer to give different results with different methods. Playing with the optimization heuristic parameters may sometimes result in drastic changes in speed and result.

APPENDIX B

USAGE OF DRCSP

B.1 Introduction

Dynamic routing control simulation program is the main tool used to test the dynamic routing control optimization methods discussed in this paper. As a startup let's look at the snapshot of the program executing a simulation on a 12-node network map.

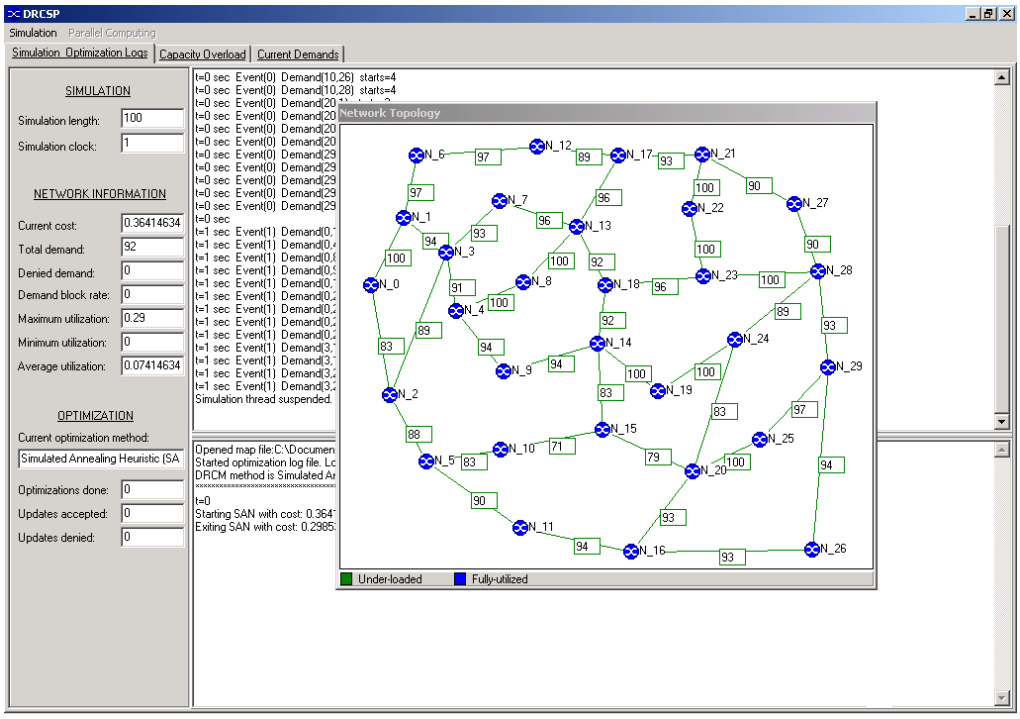


Figure 20: DRCSP.

As seen in the figure, the main program executes simulation and optimization processes on a given network map with given configuration parameters. On this figure, network map, and the optimization configuration seen on the main form are of more importance, while there are some other invisible parameters that must be taken care of. In this section, the usage of the DRCSP software GUI module is focused on and the usage of each configuration tool is told. The main configuration tools are located under the following titles:

- Network map configuration,
- Optimization properties configuration,
- Simulation properties configuration,
- Parallel processing configuration.

DRCSP software depends on a slave program if the DRCSP is programmed to use parallel processing. The program "Slave" talks with the DRCSP through windows sockets in the network. The aim of using slave programs is decreasing the processing power consumption on the DRCSP optimization module, and making the optimization process faster. Slave program does not have any user controls but a log for following the things executed. Below figure shows the "Slave" program.

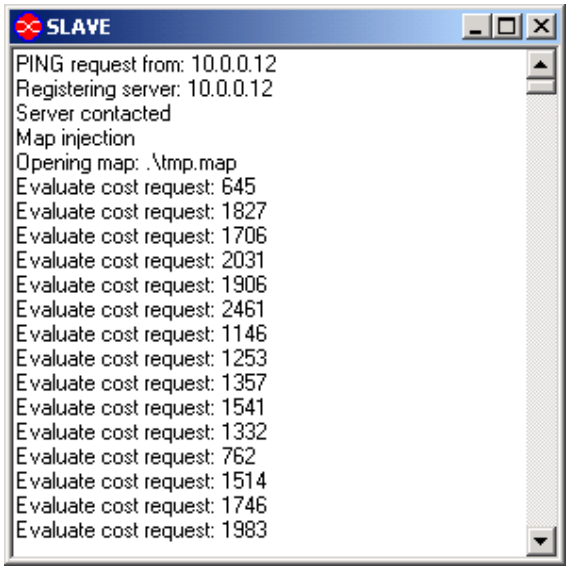


Figure 21: Slave program.

DRCSP contains an embedded network map editor, which is responsible for network map configuration. User can build a network through this tool and save it, or open an existing network map to edit it. This tool allows the user to:

- Create nodes on the map with configurable traffic generation properties,
- Connect the created nodes and configure the links with adjustable bandwidth and delay parameters,

- Create an initial inter-nodes demands table, which will guarantee an average demand between nodes,
- Delete nodes and links, change node coordinates,
- Open existing maps, edit them,
- Save modified maps.

Optimization properties configuration tool can be used to adjust optimization related parameters of the dynamic routing control method used in optimization process. This tool allows the user to:

- Select optimization method among the choices given in previous section,
- Adjust the optimization method parameters such as epoch value, population size, etc.

Simulation properties configuration is used to set the length of the simulation process and the network cost probe period for the log file of the simulation. User can force the simulation process to save the network cost periodically at the beginning of every probe period.

Parallel processing is used to speed up the genetic algorithms used in the optimization process. Genetic algorithms can be paralleled by two methods where either slave genetic algorithm processes work in parallel for a master node (distributed execution), or one instance of genetic algorithm process executes at a master node, which forces slave nodes to finish some heavy processes like fitness value evaluation. DRCSP software currently uses the second implementation, where slaves are used for cost evaluation. It cannot be said that parallel processing command set of the DRCSP is a configuration tool for the main program. In fact, parallel processing command set is used to configure the slave processes executing on the slave nodes. By the help of the parallel processing command set, user can:

- Find slave nodes by pinging,
- Inject the current network map to the slave nodes for cost evaluation purposes. Slave processes need to know the connection matrix (link connections and weight system) and the demands matrix to calculate the network costs for given GA strings.

In order to run a simulation and see the dynamic routing control methods, user has to build a network map, adjust simulation and optimization process parameters and then start the simulation. In the next subsections, mentioned tools will be explained and user will finally be able to use the DRCSP easily.

B.2 Building a Network, Creating Network Map Files

DRCSP allow the user to create or modify network map files for simulation purposes. You can create or modify a map if a simulation process is not running at that moment. While a simulation is running, opened map file is read-only and used for visualizing network link conditions. Let's create a map file step by step:

- Open the map editor: On the DRCSP Simulation menu click on "Show map"

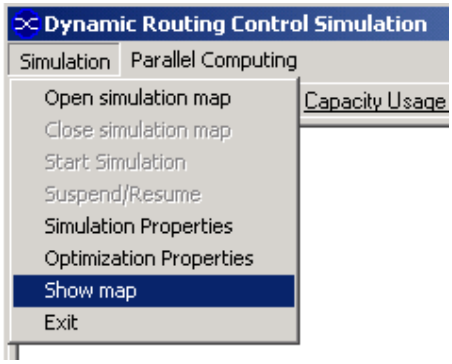


Figure 22: Showing map.

- Create nodes: Map editor becomes visible, right click on the map and select "Add node" in the popup menu.

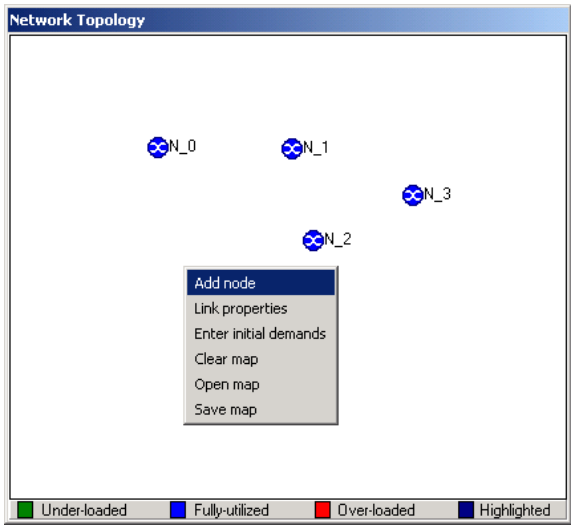


Figure 23: Adding a node.

- Delete nodes: You can delete any created node by right clicking on the desired node and selecting “Delete node”.

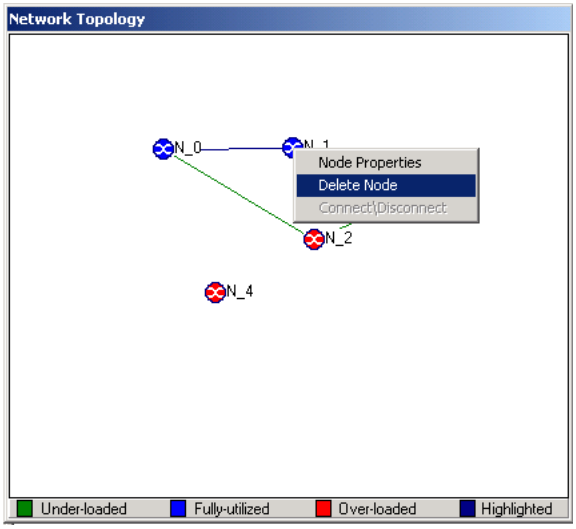


Figure 24: Deleting a node.

- Create links: Left click on two nodes, so that both nodes will be red. Right click on any of the red nodes and click on “Connect/Disconnect”.

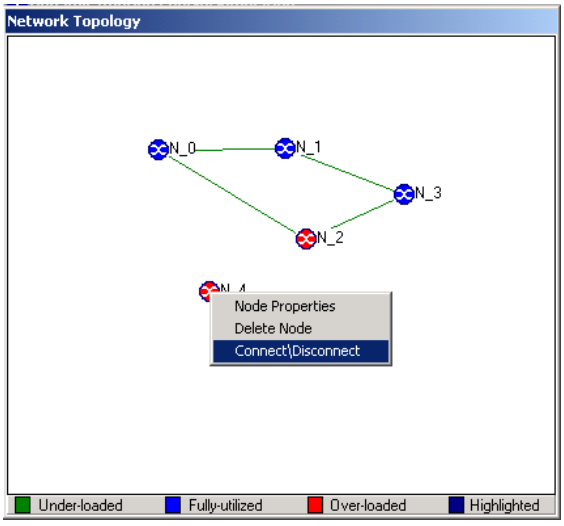


Figure 25: Connecting two nodes by a link.

- Delete links: You can disconnect a link by left clicking the end points and clicking "Connect/Disconnect".

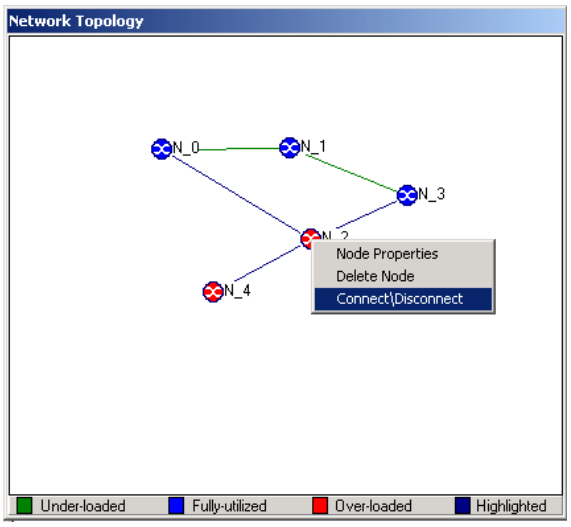


Figure 26: Disconnecting a link.

- Modify every node and configure the traffic properties: Right click on the desired node and select "Node properties".

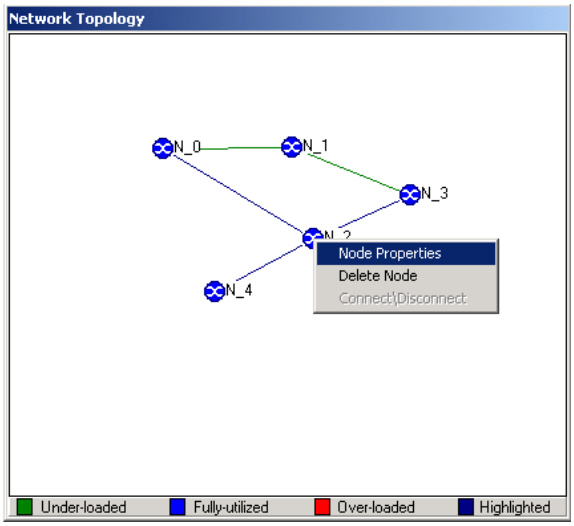


Figure 27: Opening node properties form.

Node properties dialog becomes visible, change the values as needed and click on "Set" to save and close or "Close" to quit

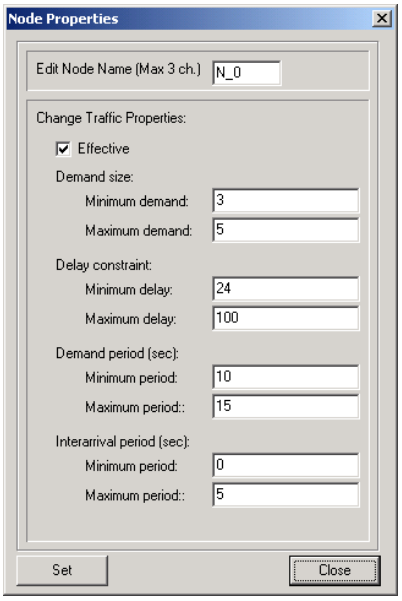


Figure 28: Node properties form.

- Change link properties: Right click on an empty place on the map and select "Link properties".

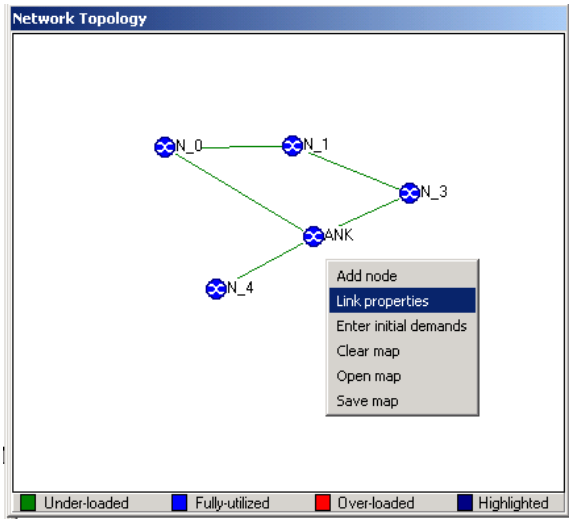


Figure 29: Opening link properties form.

Link properties dialog becomes visible, select two connected nodes from the "Select node pair" part, change the link bandwidth properties, and click on "Set" button every time you change a link's properties. Any change will not be applied if you don't click on the "Set" button. If "Node 1" and "Node 2" are not connected, you will not be able to change the link properties. After changing desired links' properties click on "Close" button to close the dialog.

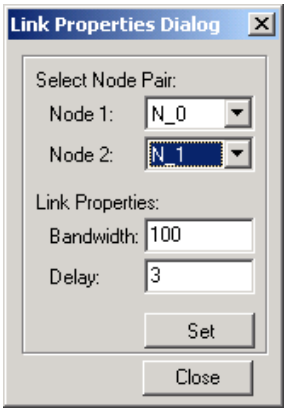


Figure 30: Link properties form.

- Enter initial demands: Right click on an empty place on the map and select "Enter initial demands".

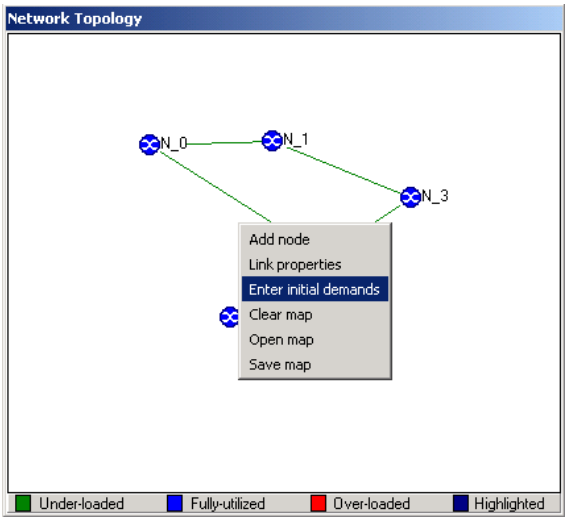


Figure 31: Opening initial demands dialog.

Initial Demands Dialog" becomes visible. Edit each initial demand value on the matrix and click "OK" to save and close or "Cancel" to close without saving. Those demands added to the matrix will be static throughout the simulation, and if dynamic demand creation is enabled on the nodes, dynamic demands will be added to those values.

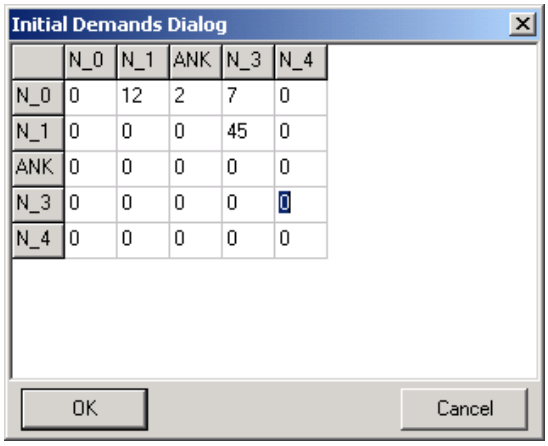


Figure 32: Entering initial demands.

- Save the map: Right click on an empty place on the map and select "Save map".

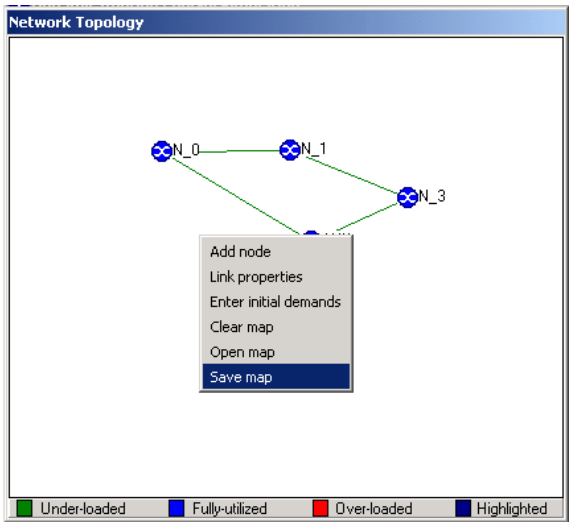


Figure 33: Saving map.

"Save Map As" dialog becomes visible. Give a name to your map file and click on "Save" button. You can overwrite any other map file, thus be careful on the warning messages.

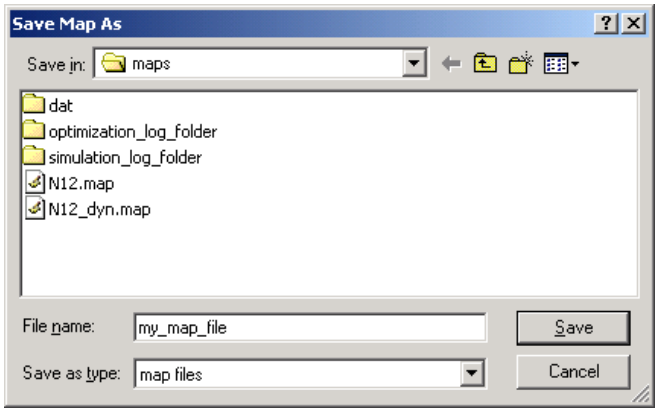


Figure 34: Save map dialog.

A map file is created successfully, but there are some other steps to clear a map, or open an existing map.

- Clear the map: Right click on an empty place on the map and select "Clear map".

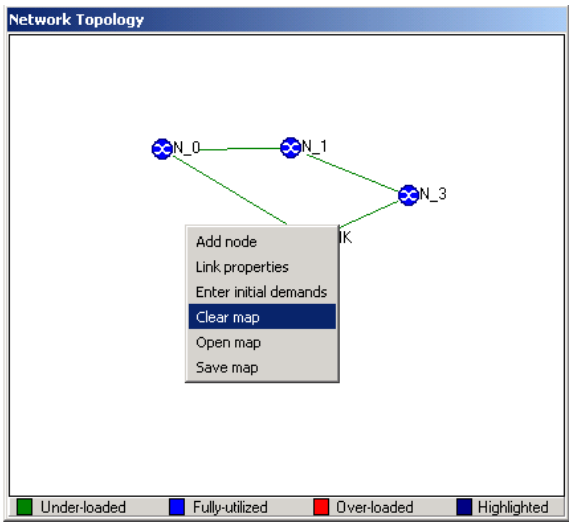


Figure 35: Clearing map.

- Open map: Right click on an empty place on the map and select “Open map”.

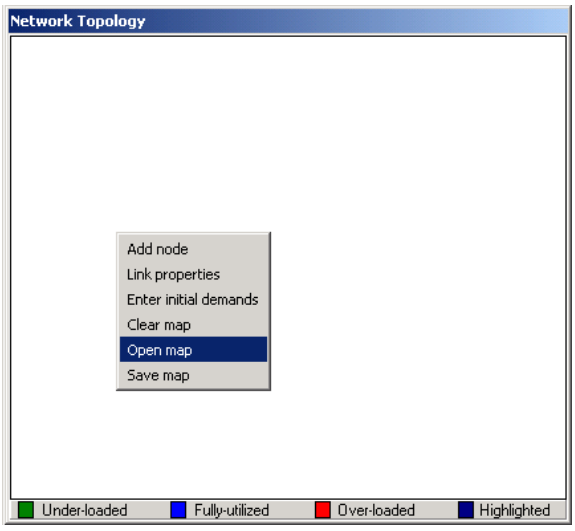


Figure 36: Opening a map for editing.

“Open Map” dialog becomes visible. Select a map and click on “Open” button to open a map file for editing.

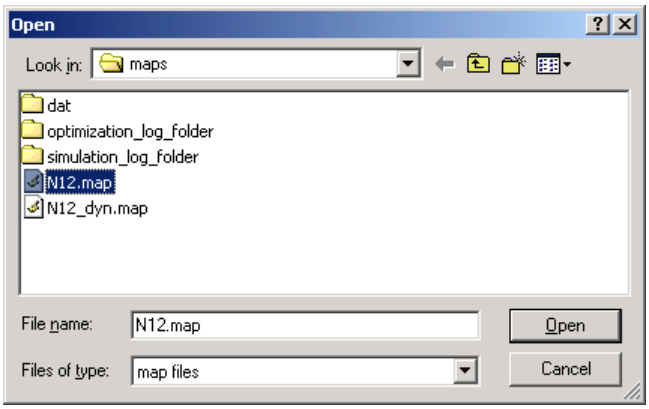


Figure 37: Open map dialog.

As the basic steps for creating a network map are carried out, let’s go on explaining the steps for using the DRCSP for simulation and optimization.

B.3. Configuring and Starting a Simulation

As discussed before, DRCSP takes .map files as input files. When a .map file is opened, DRCSP first reads the topology information including nodes, connection matrix, initial demands matrix and etc. After opening the map user has to configure the simulation properties, optimization properties and follow some extra steps for P-GA parallel processing. The steps are as follows:

- Open simulation map: On the DRCSP Simulation menu click on "Open simulation map".

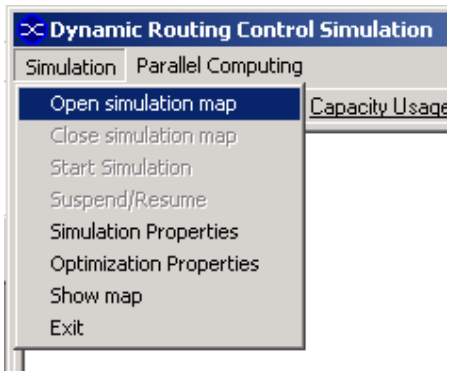


Figure 38: Opening simulation map.

"Open Map" dialog becomes visible, select a map and click on "Open" button.

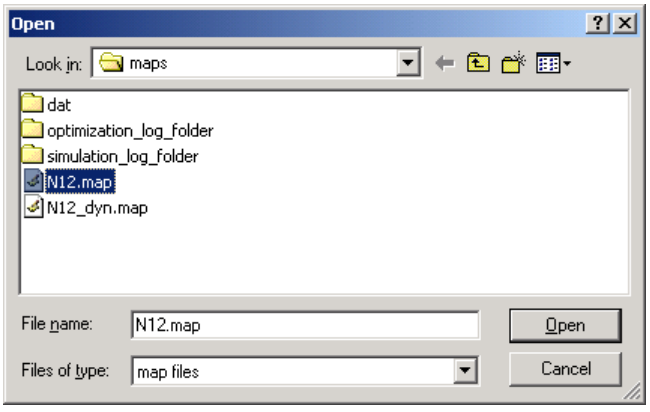


Figure 39: Open simulation map dialog.

"Map" form becomes visible, but since the map is opened for simulation, map is read-only now.

- Close simulation map: On the DRCSP Simulation menu click on "Close simulation map". User can open another simulation map file if he closes the current one.

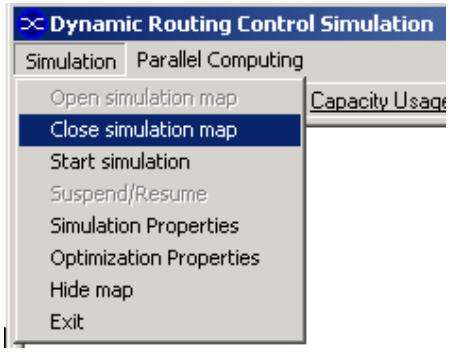


Figure 40: Closing simulation map.

- Adjust the simulation properties: On the DRCSP Simulation menu click on "Simulation Properties".

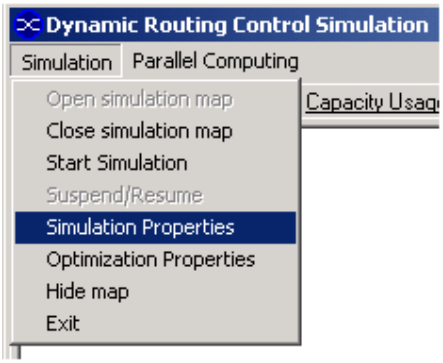


Figure 41: Opening simulation properties dialog.

"Simulation Properties" form becomes visible. Change the simulation length or cost sampling period, click on "SET" to save and close or "Close" to quit without saving.

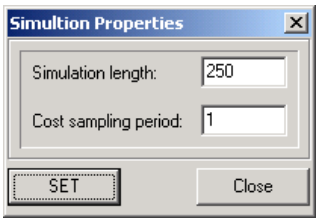


Figure 42: Simulation properties dialog.

- Adjust the optimization properties: On the DRCSP Simulation menu click on "Optimization Properties".

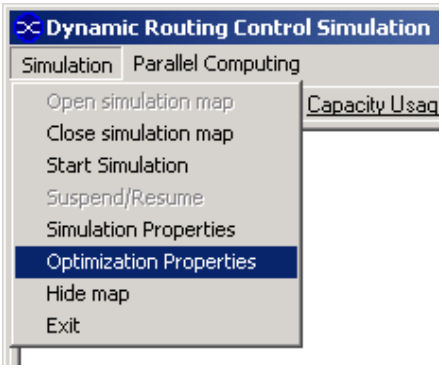


Figure 43: Opening optimization properties dialog.

"Optimization Properties" form becomes visible. Change optimization method, and optimization algorithm's parameters. Click on "Set" to save and close or "Close" to quit without saving.

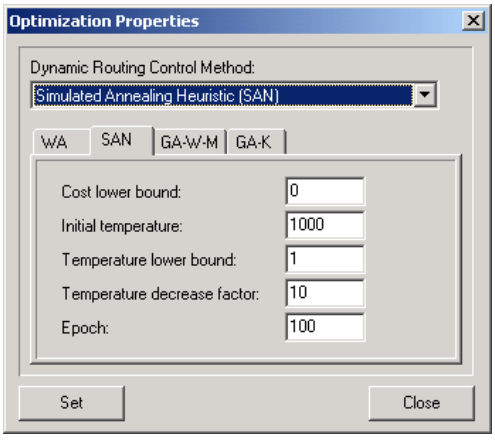


Figure 44: Optimization properties dialog.

Based on the optimization method user can directly start the simulation. If user selects P-GA method (parallel genetic algorithm) he has to go on extra steps to enable parallel processing.

- Parallel processing steps: User has to find the slave processors and inject the current network configuration to the slaves.

  On the DRCSP "Parallel Computing" menu, click on "Parallel processing" and then "Find slaves".

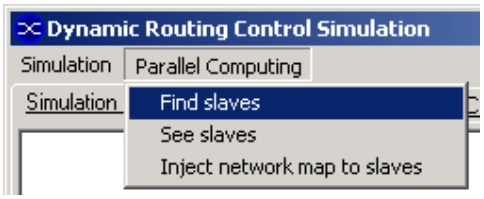


Figure 45: Finding slave machines.

On the DRCSP "Parallel Computing" menu, click on "Parallel processing" and then "Inject network map to slaves".

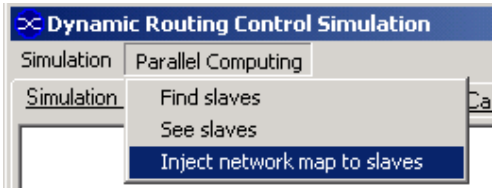


Figure 46: Injecting simulation map to slave machines.

On the DRCSP "Parallel Computing" menu, you can check for the slave processors by clicking on "See slave" at anytime.

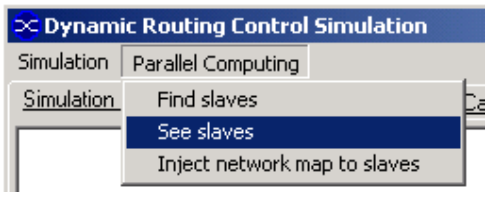


Figure 47: Seeing slave machines.

Below figure shows DRCSP and Slave programs running on the same computer. This snapshot is taken after Parallel processing steps.

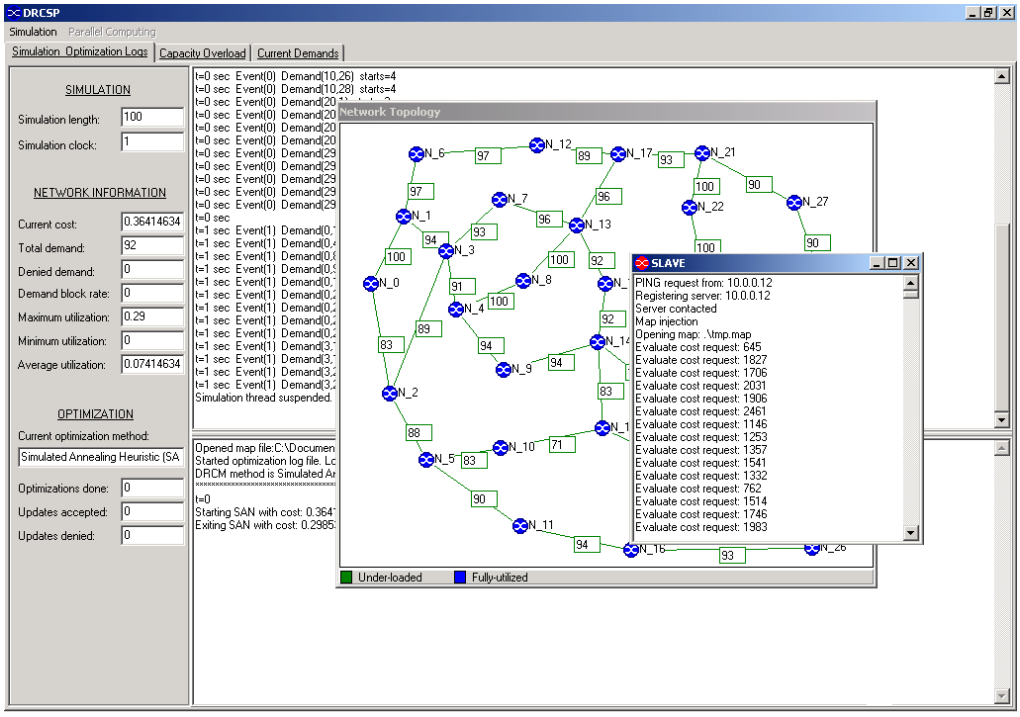


Figure 48 : Slave process and DRCSP.

- Start the simulation: On the DRCSP "Simulation" menu click on "Start simulation".

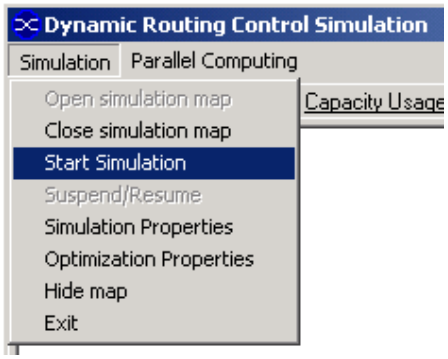


Figure 49: Starting simulation.

- Stop the simulation: On the DRCSP "Simulation" menu click on "Stop simulation".

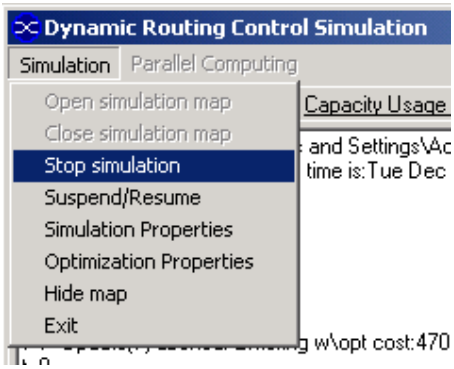


Figure 50: Stopping simulation.

- Suspend or resume the simulation: On the DRCSP "Simulation" menu click on "Suspend/Resume".

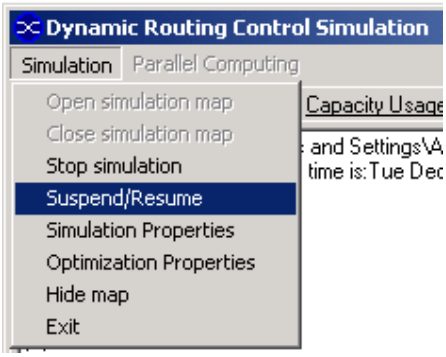


Figure 51: Suspending/resuming simulation.

The basic steps to start a simulation on map file are made. Whenever the simulation is over, DRCSP saves the simulation and optimization log files into the directory that corresponding map file exists. Optimization, simulation and network cost log files are text formatted. User can open the network cost file with a text editor, copy all lines and paste into excel sheets, so he can draw the cost vs. time graphs and see the difference between optimization methods.

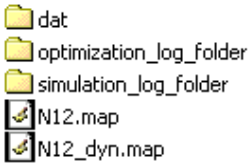


Figure 52: Log directories.