A Framework for Service Differentiation and Optimization in Multi-hop Wireless Networks

Pradeepkumar Mani

Submitted to the Department of Electrical Engineering & Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Committee:

Dr. David W. Petr (Chairperson)

Dr. Victor S. Frost

Dr. Joseph B. Evans

Dr. Man C. Kong

Dr. Erik S. Van Vleck

Date Defended

The Dissertation Committee for Pradeepkumar Mani certifies that this is the approved version of the following dissertation:

A Framework for Service Differentiation and Optimization in Multi-hop Wireless Networks

Committee:

Chairperson

Date Approved

Abstract

In resource-constrained networks such as multi-hop wireless networks (MHWNs), service differentiation algorithms designed to address end users' interests (e.g. user satisfaction, QoS, etc.) should also consider efficient utilization of the scarce network resources in order to maximize the network's interests (e.g. revenue). For this very reason, service differentiation in MHWNs is quite different from the wired network scenario. We propose a service differentiation tool called the "Investment Function", which essentially captures the network's cumulative resource investment in a given packet at a given time. This investment value can be used by the network algorithm to implement specific service differentiation principles. As proof-of-concept, we use the investment function to improve fairness among simultaneous flows that traverse varying number of hops in a MHWN (multihop flow fairness). However, to attain the optimal value of a specific service differentiation objective, optimal service differentiation and investment function parameters may need to be computed.

The optimal parameters can be computed by casting the service differentiation problem as a network flow problem in MHWNs, with the goal of optimizing the service differentiation objective. The capacity constraints for these problems require knowledge of the adjacent-node interference values, and constructing these constraints could be very expensive based on the transmission scheduling scheme used. As a result, even formulating the optimization problem may take unacceptable computational effort or memory or both. Under optimal scheduling, the adjacent node interference values (and thus the capacity constraints) are not only very expensive to compute, but also cannot be expressed in polynomial form. Therefore, existing optimization techniques cannot be directly applied to solve optimization problems in MHWNs.

To develop an efficient optimization framework, we first model the MHWN as a Unit Disk Graph (UDG). The optimal transmission schedule in the MHWN is related to the chromatic number of the UDG, which is very expensive to compute. However, the clique number, which is a lower bound on the chromatic number, can be computed in polynomial time in UDGs. Through an empirical study, we obtain tighter bounds on the ratio of the chromatic number to clique number in UDGs, which enables us to leverage existing polynomial time clique-discovery algorithms to compute very close approximations to the chromatic number value. This approximation not only allows us to quickly formulate the capacity constraints in polynomial form, but also allows us to significantly deviate from the traditional approach of discovering all or most of the constraints *a priori*; instead, we can discover the constraints as needed. We have integrated this approach of constraint-discovery into an active-set optimization algorithm (Gradient Projection method) to solve network flow problems in multi-hop wireless networks. Our results show significant memory and computational savings when compared to existing methods.

Acknowledgements

I would first like to thank Dr. David W. Petr, Dr. Victor S. Frost, Dr. Joseph B. Evans, Dr. Man C. Kong, and Dr. Erik S. Van Vleck for serving on my committee, and for agreeing to meet for my dissertation defense on short notice. I wish to express my heartfelt gratitude towards my advisor Dr. Petr for taking me under his wings and mentoring me for the last seven years. He ensured that I received financial support throughout my graduate study and provided sufficient latitude in the choice and topic of my research, without which this dissertation would not have been possible. He constantly encouraged new research ideas, and provided support and guidance to refine those ideas, which helped in the timely completion of this dissertation. He was always available to discuss various topics including (but not limited to) research, teaching, future goals, job search, etc. I consider him not just as my academic advisor, but also as a professional role-model, and look forward to continue and benefit from interactions with him in the future.

I would like to thank Dr. Frost for supervising my work on the SensorNet project. My work on the SensorNet honed my implementation and programming skills, which facilitated algorithm development and its implementation for this dissertation. Dr. Evans' courses on high-performance networking and Internet routing enabled me to develop a strong understanding for networking concepts. Dr. Kong's graph theory course was the foundation of all the graph-theoretic concepts used in this dissertation. The numerical analysis course that I took under Dr. Van Vleck helped me understand and work around some of the numerical issues that I encountered while developing the modified gradient projection method of optimization for this dissertation.

I would also like to thank all my colleagues at KU and ITTC and my friends that have made my journey through graduate school a very memorable experience.

I would also like to extend a special word of thanks to the support staff and system administrators at ITTC for their remarkable work in providing support for students' research needs. My appreciation for the system administrators increased multi-fold after my experiences with the industrial world during my internships.

Finally, I would like to thank my mom, dad, my in-laws, Satish, Pradeepa, Darshan and my dear wife Priya for all their love and support. Priya ensured that I was not distracted while working on my PhD, and has been very patient and understanding of the time spent on my research. Needless to say, without her, my life during PhD would have been very difficult.

Contents

Acceptance Page **i**

Abstract **ii**

Acknowledgements **iv**

1 Introduction **1**

1.1 Multi-hop Wireless Networks . . . 1

1.2 MHWN Preliminaries . . . 4

1.3 Service Differentiation Tool for MHWNs . . . 7

1.4 Optimization Framework for MHWNs . . . 9

1.5 Related Work . . . 11

1.6 Organization of Dissertation . . . 16

2 Investment Function **18**

2.1 Motivation . . . 18

2.2 Investment Function . . . 20

2.2.1 Sample Investment Function . . . 21

2.3 Sample Network Objective . . . 23

2.3.1 Sample Service Differentiation Algorithm . . . 24

2.4 Performance Evaluation . . . 26

2.4.1 Simulation Scenario . . . 28

2.4.2 UDP Simulations . . . 30

2.4.3 TCP Simulations . . . 33

2.5 Optimality of Results . . . 38

2.6 Summary . . . 42

3 Capacity Constraints in MHWNs **44**
3.1 Interference Modeling in MHWNs 44
3.2 Unit Disk Graph Model 48
3.2.1 Limitations of UDG model 50
3.3 Scheduling and Interference Computation 50
3.4 Graph Coloring and Interference 57
3.5 Cliques and Clique Number 60
3.6 Imperfection Ratio 62
3.7 Summary 63

4 Bounds for Practical MHWNs **65**
4.1 Practical Bounds on Imperfection Ratio 65
4.2 Consequences of Practical Bound 71
4.3 Summary 80

5 Optimization Framework for MHWNs **81**
5.1 Network Flow Problems in MHWNs 81
5.2 Discover-as-you-go Approach ($DAYG$) 85
5.3 Modified Gradient Projection Method 87
5.4 Evaluation 90
5.4.1 Grid Topology 95
5.4.2 Random Topology 98
5.4.3 Miscellaneous 101
5.5 Summary 104

6 Conclusions and Future Work **105**
6.1 Conclusions 105
6.2 Future Work 107

Bibliography **109**

List of Figures

1.1 Brick-Wall model of Communication and Carrier-Sensing 6

1.2 Example showing Transmission Range and Carrier-Sense Range of node u. Node v is within T_R of node u, while nodes v, w, and x are within C_R of node u 7

2.1 Dynamic Priority Mapping in a 3-Level Priority Queue 26

2.2 4x4 Grid Topology 29

2.3 *pdr* and mean delay plots for UDP-CU 31

2.4 *pdr* and mean delay plots for UDP-VU 32

2.5 $UDP - VU$: Mean Delay Vs. Priority 33

2.6 TCP: Mean Delay Vs. Hop Count 36

2.7 TCP: Mean Delay Vs. Hop Count 38

2.8 $TCP - VU$: Priority Results 38

2.9 LP formulation for MAXFLOW between source r and destination s in a wired network 41

3.1 Example showing the conservative nature of the Tx-model of interference 46

3.2 Chain Topology of 6 nodes. Inter-node distance ($d(u, v) = 200m$) 49

3.3 Communication UDG for 6-node chain topology. $T_R = 250m$ . . . 49

3.4 Interference UDG for 6-node chain topology. $I_R = 500m$. 50

3.5 6-node chain topology augmented with traffic rates. Each node transmits at the rate of 1 unit per second on every outgoing link . 55

3.6 Weighted Interference UDG for 6-node chain topology. Weights are enclosed within circles. A node's weight corresponds to it's total outgoing transmission rate. 56

3.7 Pessimistic and Optimal scheduling. The scheduled node indices are shown within rectangles 57
3.8 For a sample UDG (G), transformation from a weighted graph to unweighted graph (H), followed by optimal coloring. Three colors are used: *Red* (R), *Green* (G) and *Yellow* (Y) 58
3.9 Example weighted graph to illustrate clique and maximal clique concepts. A total of 9 cliques can be identified: $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1,2\}$, $\{1,3\}$, $\{2,3\}$, $\{3,4\}$, $\{1,2,3\}$. Out of these 9 cliques, only two are maximal cliques: $Q_1 = \{1,2,3\}, Q_2 = \{3,4\}$. 61

4.1 CDF plots for Partial Imperfection Ratio (PIR) for various *(n,K)* values and $\beta = 1$ 68
4.2 95 and 99 percentile PIR values for various *(n,K)* values and $\beta = 1$ 69
4.3 CDF plots for Partial Imperfection Ratio (PIR) for various (β, k) values for $n = 100$ 70
4.4 95 and 99 percentile PIR values for various (β, k) values for $n = 100$ 71
4.5 Reduced Interference UDG corresponding to node 4 (G_4^R) in 6-node chain topology. 73

5.1 LP formulation for MAXFLOW in MHWNs 82
5.2 Performance Plots for Grid Topology 96
5.3 Performance Plots for Random Topology 100
5.4 Memory Requirements for Random Topology 103

List of Tables

2.1 Mean and Variance of pdr and Delay for $UDP - CU$ 30
2.2 $UDP - CU$: Network Investment 31
2.3 $UDP - VU$: Network Investment Results 32
2.4 $UDP - VU$: Fairness Results 33
2.5 Throughput for $TCP - CU$ 35
2.6 $TCP - CU$: Fairness Results 36
2.7 Throughput for $TCP - VU$ 37
2.8 $TCP - VU$: Fairness Results 37

3.1 Communication neighbors and Interfering neighbors for 6-node chain topology 50

4.1 Maximum Weighted Cliques for each node in the 6-node chain topology 78

5.1 Average Number of Super-Maximal Cliques in $CLIQUDG$ for the Random Topology 101

Chapter 1

Introduction

1.1 Multi-hop Wireless Networks

Multi-hop networks are networks where the communicating entities (nodes) in the network are not directly "connected" to one another. Transfer of data between any two nodes could potentially pass through multiple intermediate nodes (hops) before reaching the intended destination. Multi-hop networks can be classified based on the medium of communication:

- *wired*, where the communication between the nodes is realized through a wired medium (e.g. traditional Internet)
- *wireless*, where the communication medium is wireless
- *hybrid*, where the network consists of a good mix of nodes using both wireless and wired media for communication

In this dissertation, we focus only on multi-hop *wireless* networks (MHWNs). Multi-hop wireless networks find application in a number of environments; they can provide an alternative to a last-mile wireline infrastructure in geographically infeasible areas, can serve as limited-lifetime networks that provide communications infrastructure for disaster response (e.g., hurricane Katrina) or for security for infrequent, large-audience events (e.g., NASCAR races), and can provide communications for networks of sensors. MHWNs can be differentiated based on whether the nodes in the network are mobile or static:

- *Mobile Ad hoc NETworks* or MANETs [man] are MHWNs in which the nodes in the network are capable of mobility. Usually, these networks are characterized by lack of infrastructure as well (and thus "ad hoc"). Typical application scenarios include emergency and disaster response situations.
- *Wireless Mesh Networks* are MHWNs in which the nodes are static. These networks are usually ad hoc as well. Perhaps the most popular network in this category is the wireless sensor network.

The classification presented above is not rigid, and is made for illustration purposes only. Networks that do not fall under either category are possible. One could have a set of mobile users communicating with one another using a mesh network of wireless access points. Various networking technologies and hardware have been used to build the MHWNs. Some popular wireless networking standards used in MHWNs are:

- *PHY layer*: 802.11 a/b/g/n [80207], 802.15.4 [80206]
- *MAC layer*: 802.11, 802.11e [80207], 802.15.4 [80206]
- *Networking architectures*: ZigBee [zig], 6loWPAN [KMS]
- *Hardware*: mica motes [mot], SunSPOTs [sun], GumStix [gum]

For a complete survey on the various technologies used in the various layers of the wireless networking nodes, see [AWW05]. While the various categories of MHWNs could vary in terms of specific networking technology used, application scenarios, communication paradigms, etc., there are some unifying characteristics:

- Resource constraints on wireless channel bandwidth, node processing power, electric power supply, etc. Usually, the nodes in MHWN are stand-alone nodes that operate on battery power. So electric power is a very important constraint in MHWNs.
- Lack of infrastructure
- Relatively small network size
- Adjacent-node interference that arises due to omni-directional broadcast transmission over a shared medium.

Much of the work in the dissertation revolves either directly or indirectly around the adjacent-node interference problem. While the interference problem is prevalent in *any* wireless network, it is exacerbated in the multi-hop case due to contention from packets belonging to the same flow but at different hops, leading to extremely biased and inefficient bandwidth utilization. For the particular case of 802.11 DCF, it was shown in [LBD$^+$01] that the throughput of a single flow traversing a chain of four or more wireless hops is upper-bounded by 0.25 of the throughput attainable if the flow traverses only one hop, with actual reduction factors closer to 0.14. When multiple flows are present, the situation deteriorates even further, with strong dependencies on traffic patterns. Note that this interference problem is absent in wired networks, and it can be easily controlled in single-hop wireless or cellular networks using a centralized scheduler.

1.2 MHWN Preliminaries

Assume that we are given a MHWN of n nodes $(1, 2, .., n)$, and their respective position coordinates in two dimensional space. At the physical layer, the ability of the wireless transceiver to successfully receive information is directly proportional to the *signal-to-noise ratio* or the SNR value [Pro00] measured at the receiver. The SNR is simply the ratio of received signal power(P^R) to the ambient *noise* power (N_p). The received power value decreases as a non-linear function of distance between the sender and the receiver nodes for a given transmit power P^T. These models factor in *large-scale fading* [Rap01] or attenuation and sometimes *small-scale fading* [Rap01]as well. Once the received power is computed using these models, the SNR is then computed by assuming a constant value for ambient noise power. The receiver characteristics will usually include a "receiver sensitivity" ($RxThresh$) parameter, which is the minimum required received signal power, such that for a given noise power, the SNR at the receiver is greater than (SNR_{th}), so that the probability of error in receiving the transmitted bit is very small.

$$P^R > RxThresh => \frac{P^R}{N_p} > SNR_{th} => P(error) < \epsilon, \epsilon \to 0$$

In CSMA networks (for e.g. 802.11), a given node checks to see if the carrier can be "sensed" (i.e. checks to see if the carrier is busy) before engaging in any transmission activity. The channel will be "sensed" by a given node v if there is a transmission by some node u, such that the received power at v due to u's transmission exceeds the *carrier-sense threshold* or $CSThresh$ ($P^R > CSThresh$). The importance of carrier-sensing can be illustrated as follows: if node u transmits to node v, then a simultaneous transmission from node w to some other node x can corrupt the reception at node v if node v can sense the carrier due to node w (the received "interference" or "noise" power at node v due to node w's transmission exceeds $CSThresh$).

The nodes in a CSMA network usually defer transmission if the carrier due to an unrelated transmission can be sensed. $CSThresh$ is a tunable parameter that was devised as a means to overcome (albeit, unsuccessfully) the *hidden* and *exposed* node problems [All, BDSZ94] that are inherent to wireless networks. Usually, the carrier-sense threshold is set lower than the receiver sensitivity ($CSThresh < RxThresh$). Some mechanisms of tuning the $CSThresh$ can be found in [YYS03] and [DLV04].

Very often, for ease of analysis, the physical layer details are abstracted, and analytical models are built using simplifying assumptions. These models are *binary* or *brickwall* models of communication (carrier-sensing), where a node can communicate (sense the carrier) with probability 1 if the receiver (sender) lies within a distance of T_R (C_R), and with probability 0 otherwise. If interpreted in terms of received power, the binary models imply that the probability of successful reception is 1 if $P^R \geq RxThresh$, 0 otherwise, and the probability of successful carrier sensing is 1 is $P^R \geq CSThresh$, 0 otherwise. Figure 1.1 shows the binary model using received power values. This is also the model built into the ns-2 simulator [ns2] that is very widely used in the research community. We adopt a similar model later in this dissertation when we use a graph-theoretic model to represent MHWNs.

Based on brick-wall model, the following distance parameters can be defined:

- *Transmission range*: The *Transmission range* (T_R) of a given node is de-

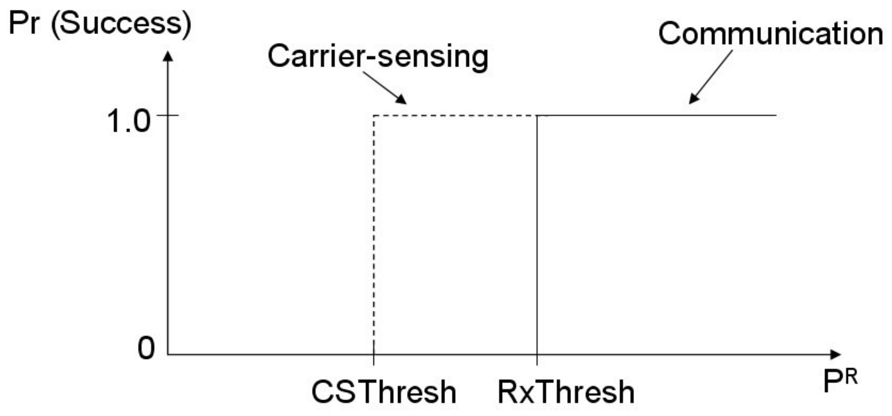


Figure 1.1. Brick-Wall model of Communication and Carrier-Sensing

fined as the maximum distance at which the node's transmission can be successfully received without corruption, assuming that there are no other transmitters in the vicinity of the receiver (thus, corruption occurs only due to ambient noise). Hence, T_R can be interpreted as the maximum distance beyond which the received signal power $P^R < RxThresh$, for given physical layer model. In other words, T_R is the maximum distance computed using a given physical layer model, beyond which the measured $SNR < SNR_{th}$. Thus, the sender node can communicate directly with the receiver node only if the receiver lies within a distance of T_R from the sender.

- *Carrier-sense range*: *Carrier-sense range* (C_R) of given node is defined as the maximum distance such that the transmission by this node will be received with $P^R \geq CSThresh$. It is the threshold distance beyond which a given node's transmission (or carrier) cannot be "felt" or sensed.

If $CSThresh < RxThresh$, then $C_R > T_R$. Fig.1.2 shows an example scenario to illustrate the transmission and carrier-sense range concepts.

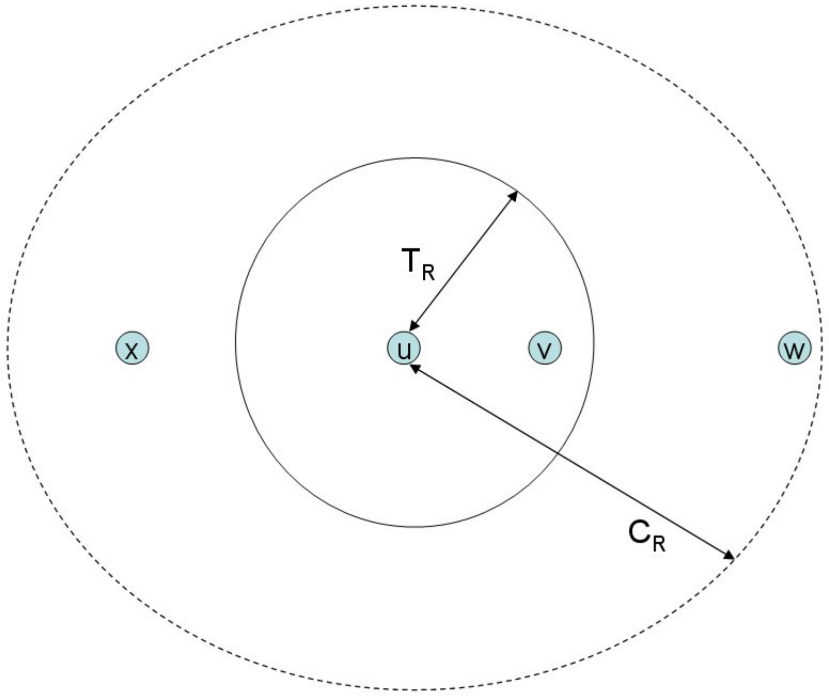


Figure 1.2. Example showing Transmission Range and Carrier-Sense Range of node u. Node v is within T_R of node u, while nodes v, w, and x are within C_R of node u

1.3 Service Differentiation Tool for MHWNs

Due to the ill-effects of adjacent-node interference, efficient resource utilization becomes a critical issue in MHWNs. Service differentiation may be required to achieve certain network objectives (e.g. QoS) in MHWNs. Service differentiation can be defined as the process of providing better treatment (higher access to resources) to relatively more “important” traffic flows when compared to less important ones. Identifying and quantifying “importance” of packets (and thus flows) is a very critical part of service differentiation. In wired networks, usually the nature of traffic (real-time, non-real time, etc) and monetary considerations (more importance to higher paying users) influence the importance accorded to flows. Usually, more important (higher priority) flows are provided better treatment at the expense of less important (lower priority) flows; in extreme cases it may involve starving or dropping packets belonging to lower priority flows.

However, in MHWNs, given the scarcity of network resources, we argue that the extent of network resources expended for a given flow must also be factored into quantifying and assigning importance. *We emphasize that we do not develop a new service differentiation architecture; instead we develop a tool called the "Investment Function" that aids in assessing the relative importance of a given packet.* The investment function factors in prior network investment into the packet, in addition to monetary considerations. This information can then be used by the service differentiation algorithm to realize a given service differentiation objective. Although we believe the investment function potentially has very broad networking applications, we will concentrate on its applicability in a multi-hop wireless network context, where we believe its contributions can be substantial.

We demonstrate the effectiveness of the investment function by applying it to a sample network objective of enhancing multi-hop fairness and efficient utilization of the scarce bandwidth in multi-hop wireless networks. We develop a simple heuristic service differentiation algorithm that makes use of the investment function to achieve a two-pronged objective: significant increases in network bandwidth utilization, while allocating and distributing the bandwidth among flows to promote service quality and ensure fairness among flows. Our simulation results show significant improvement in multi-hop flow fairness for both TCP and UDP flows, while simultaneously reducing resource wastage when using the investment function. This validates the usefulness of the investment function in a MHWN scenario.

1.4 Optimization Framework for MHWNs

While our sample network objective of enhancing multi-hop flow fairness and network utilization efficiency showed improvement when the investment function was used, the results were by no means optimal. This is because the definition of the investment function and the service differentiation algorithm were based on heuristics. To design investment function and service differentiation algorithms to achieve optimal network objectives, suitable parameters for these algorithms need to be computed. This can usually be achieved by formulating a network flow problem for the given MHWN to optimize the desired network objective, with the algorithmic parameters suitably captured in the constraints of the optimization problem.

Unfortunately, the formulation of the optimization problem is very difficult for MHWNs. This is due to the difficulty in formulating the *capacity constraints*, which involves capturing the interference, and expressing the interference value in polynomial form. The interference sensed by the nodes in the network is directly related to the underlying scheduling scheme used to schedule transmissions of the various nodes. Any scheduling scheme used for MHWNs should be exploit the possibility of spatial reuse of the carrier frequency in a MHWN in order to increase throughput. An optimal scheduling scheme is one that maximizes the network throughput for a given set of traffic rates of the various nodes. If optimal scheduling is assumed, the schedule can then be computed using graph-theoretic techniques, specifically graph coloring. However, for most graphs, the optimal coloring process is computationally prohibitive, and in most cases, one has to be satisfied with approximations or bounds on the optimal coloring number.

The clique number of any graph lower bounds the chromatic or coloring number. The clique number is as difficult to compute as the chromatic number for most classes of graphs. However, for a special class of graph called the Unit Disk Graph (UDG) [BCD90], the clique number can be computed in polynomial time, as shown in [BCD90]. Fortunately, a MHWN lends itself naturally to be modeled using a UDG. Prior research has shown that the ratio of the chromatic number to clique number can be bounded within a factor of 2.155 [GM01]. In this dissertation, we show through an empirical study that a much tigher bound can be used for the ratio of chromatic number to clique number for practical MHWN-related UDGs.

Based on our empirical results, excellent approximations to a particular constraint can be computed in polynomial time under UDG formulation (clique number). The bad news is that there could be a potentially exponential number of constraints (cliques) for a given UDG [GWG05]. Traditional approaches of formulating the optimization problem have included finding a large subset of constraints [JPPQ03] or good approximations of the constraints (cliques), such as the clique-generation approach in [GW04] or listing all *super-maximal cliques* in [GWG05]. These approaches have steep memory requirements and tend to be computationally expensive both at the problem formulation phase (due to finding all or many cliques), and at the problem solving phase (due to the scale of the problem with a large number of constraints).

To address this problem, we develop a *discover-as-you-go* approach, in which the constraints are discovered as and when required by the optimization solution algorithm. This is in contrast to the traditional methods that enumerate all constraints during problem formulation. Our approach has its inspiration from the *active-set theorem* [Lue84] in optimization theory. We integrate this approach of finding constraints with a popular active-set strategy (Rosen's Gradient Projection Method [Ros60]) for solving optimization problems. We present significant memory and run-time savings using our method when compared to the existing strategies. We also show that the total number of constraints needed in the discover-as-you-go approach is significantly lower than the number required by traditional methods.

1.5 Related Work

MHWNs are plagued with bandwidth utilization and fairness problems. The problem of poor transport layer performance in wireless networks has been attributed to various factors such as mobility, erroneous congestion control, contention of TCP packets with ACK packets, link-layer contention (lack of bandwidth), etc. [RS05, GTB99, FZL+03, XPMS01]. Xu et. al. in [XS01] cite an example where a 1-hop TCP flow completely shuts down a simultaneously active 2-hop TCP flow in its neighborhood. To improve end-to-end throughput, numerous localized solutions have been proposed such as tweaking of TCP parameters, modifying 802.11 DCF, modified link-layer schemes, drastic changes in TCP architecture, etc [RS05, GTB99, FZL+03, NKGB00, LBS99, SAHS05]. We strongly believe that end-to-end delivered throughput (sometimes called application goodput) can be substantially improved by maximizing network utilization efficiency. To our knowledge, we are the first to explore this avenue. The unifying tool that we use to achieve this objective (and others) is the investment function.

Many service differentiation architectures such as IntServ [BCS], DiffServ [BBC+], etc. have been developed for the Internet. Various researchers have studied the feasibility of using these architectures and their variants in the context of MHWNs (see [XSLC00, LC05]). Some MHWN-specific architectures such as [LAZC00] have been developed as well, to provide QoS in MHWNs. In all of these approaches, resource allocation to the flows is based on some importance metric of the flow or packet belonging to the flow. We do not propose a new service differentiation scheme; instead we develop a novel framework which allows us to specify a specific service differentiation algorithm and a related investment function definition that computes the "importance" of each packet, without the need to maintain any flow-related state in the intermediate nodes.

The investment function that we describe has a small degree of overlap with the price-based approach discussed in [XLN06, QM03], but there are fundamental differences in terms of applicability, objective function and computation of price or investment. T.Strayer in his dissertation [Str92] develops the "importance function" concept in the general context of task scheduling, where each task is associated with an importance function that provides the importance of each task to the global system at any given time. He shows that common scheduling algorithms can be expressed via appropriate definitions of importance functions. While our investment function concept is similar to the importance function in terms of providing an importance value to the entity to be processed (packet, task), there are fundamental differences. The importance function is specific to scheduling problems, and aids in expressing scheduling algorithms in functional form. Our emphasis is on arriving at a suitable investment function definition to capture network investment to develop suitable service differentiation algorithms to achieve a given network objective. In fact, once the algorithm is developed, the importance function could conceivably be used to describe the algorithm in functional form.

To analyze the quality of the investment function definition and service differentiation algorithm, it becomes necessary to solve a network flow problem in the given MHWN. As a first step, we model the MHWN using a graph. Traditionally, a number of networking problems such as scheduling, resource allocation, optimal network flow, and minimum cost routing in the wired domain have been solved by modeling the network as a graph and applying graph algorithms on such graphs (for extensive treatment, see [AMO93, Lue84]). The wealth of graph algorithms and their relationship to networking problems is an obvious incentive to model networks as graphs. Some of these techniques have been extended to solve network problems in single-hop wireless networks as well (e.g. frequency allocation using graph coloring [Hal80] [GSW98]). The adjacent-node interference problem present in MHWNs hampers application of these well-studied methods to solve problems in the MHWN domain. Additional effort is required to suitably model the interference relationship between the various nodes in the MHWN. These interference values manifest as *capacity constraints*.

In [KN05], Kodialam et. al. derive necessary and sufficient conditions using edge coloring for feasibility of a rate vector in a multi-hop wireless mesh network with orthogonal channels. They also propose algorithms to determine the achievable rate region as an approximation to the optimal value. In [BJ08], Bazan et. al. use the work from [KN05] to formulate a multi-commodity flow problem for MHWNs with smart antennas. Though our work has some overlap with [KN05], our formulation of the capacity constraints is quite different. We also show that it is both sufficient and necessary for any rate vector to satisfy the capacity constraints to be feasible.

Various solution techniques have been proposed to solve specific instances of flow problems in MHWNs [JPPQ03,GMW07,KMPS05]. The interference problem was not addressed explicitly in these papers. In [JPPQ03, GMW07], the impact of interference was indirectly captured using cliques as approximations, while in [KMPS05], the interference value on a given link was upper-bounded using the sum of link rates of all interfering links (pessimistic scheduling). These approaches are mostly limited to the specific flow problem at hand, and not to a general flow problem. In contrast, our framework is quite general and can be used to solve generalized flow problems in MHWNs.

In [LQZ+07], the authors attempt to explicitly model interference to identify high-throughput paths while routing packets. Interference is computed using an approximation algorithm to generate cliques (related to [JPPQ03]). In [vR-SWZ05], a model of interference is presented in the context of topology control for MHWNs using UDGs. However, the interference is defined in terms of number of nodes which is not particulary useful for solving network flow problems. In [CdGB07], interference is modeled explicitly, but the total interference is computed under a pessimistic scheduling scheme (sum of transmission rates of all interfering neighbors). It is important to note that constraint formulation under a pessimistic scheduling scheme is trivial, and the performance measure obtained using a pessimistic scheduling scheme serves as a lower bound on the optimal value, and could deviate significantly from the optimum. Our explicit model of interference is quite general and allows for a variety of scheduling algorithms. In this dissertation, as a working example, we choose the optimal scheduling scheme, which is perhaps the toughest for computing interference. By using a UDG to model the given MHWN, we were able to provide tighter bounds on the interference value under an optimal scheduling scheme. The study of interference bounds is an extension of the work in [MP07].

Another major contribution in our work is the Discover-As-You-Go ($DAYG$) approach to formulate constraints where constraints are discovered on-the-fly, in contrast to the approaches ([JPPQ03, GWG05]) which discover constraints *a priori*. The key observation which motivated the $DAYG$ approach was that both [JPPQ03,GWG05] probably generate a lot more constraints than what is actually required to solve the optimization problem. The $DAYG$ approach discovers the constraints only as needed thus leading to manageable number of constraints, and lower runtime. It has to be noted that the method in [JPPQ03] is quite general and does not rely on an underlying UDG, whereas the method in [GWG05] is applicable only to UDGs. The $DAYG$ method is quite general as well, in the sense that it does not require an UDG to operate - it only requires a polynomial-time separation oracle ($LCONSTR$) for efficient runtimes. For general graphs, the separation oracle may not be polynomial-time, thus impacting the runtime efficiency of the optimization algorithm ($MGPM$). However, the $DAYG$ approach still provides significant savings in memory requirements when compared to [JPPQ03,GWG05] due to the limited number of constraints discovered.

In short, our major contributions in this dissertation can be summarized as follows:

1. Developed the concept of investment function as a tool to aid service differentiation in MHWNs

2. Demonstrated tighter bounds for the ratio of chromatic number to clique number in practical MHWN-related UDGs

3. Formulated *global capacity constraints* for network flow-related problems in

MHWNs, and showed that they are necessary and sufficient conditions for feasibility

4. Developed the notion of *local capacity constraints* for network flow-related problems in MHWNs that are easier to formulate than the global capacity constraints and can be used in place of the global capacity constraints.

5. Developed a *Discover-as-you-go* approach to discover constraints while computing optimal network objective value in MHWNs

6. Demonstrated substantial improvments in memory efficiency and speed compared to existing MHWN optimization formulation and solution approaches, while simultaneously improving the accuracy of the desired solution.

1.6 Organization of Dissertation

This dissertation has been organized as follows:

- In Chapter 2, we develop the notion of investment function, and demonstrate its usefulness via a proof-of-concept study. We also observe the difficulty in obtaining optimal service differentiation and investment function parameters.

- In Chapter 3, we introduce tools necessary to formulate flow optimization problems in MHWNs. In particular, we model the given network using Unit Disk Graphs, and introduce global capacity constraints necessary for the optimization framework.

- In Chapter 4 we demonstrate an accurate polynomial-time approximation to the ratio of chromatic number to the clique number in UDGs. We introduce the notion of local capacity constraints and demonstrate that they are quicker to formulate than the global capacity constraints, and can be used in place of the global capacity constraints by virtue of possessing properties similar to that of the global capacity constraints.

- In Chapter 5, we develop an optimization framework for MHWNs. We develop the modified gradient projection algorithm by integrating the polynomial-time constraint formulation procedure into an existing optimization algorithm (gradient projection method). Through performance analysis experiments, we demonstrate its superiority over existing methods of solving flow problems in MHWNs.

- In Chapter 6, we present the conclusions of this dissertation, along with directions for future work.

Chapter 2

Investment Function

2.1 Motivation

In networking systems, service differentiation algorithms are commonly used to dictate access to resources to packets or flows based on a specific set of policies. These policies are framed to achieve an overall system objective. As an example, quality-of-service (QoS) provisioning to flows as requested by the end users is one such system objective for which service differentiation is employed. Two popular QoS mechanisms exist in the Internet, the IntServ and the DiffServ architecture. The Intserv architecture aims to provide *hard guarantees* as requested by the user and involves resource allocation at each intermediate node via some signalling mechanism (e.g. RSVP [BZB$^+$]). The DiffServ architecture does not provide hard guarantees; instead it defines various service classes, with each class offering varying degrees of preferential treatment to packets or flows mapped to it. The mapping to a specific service class is done based on the importance or "priority" of the packet. For example, in DiffServ, the DiffServ Code Point (DSCP) present in the IP TOS field of the packet is used to map the packet to a specific class (EF, AF, etc).

Conventionally, the priority of the packet or flow depends primarily on the *user-assigned* priority. The user-assigned priority, as the name implies, is assigned by the user to indicate the level of service that the user expects to receive from the network. The level of service requested by the user (user QoS) could depend on the type of application that generates the traffic flow (real-time, non real-time, etc.). Some times, monetary considerations could also influence the priority of the packet. The traffic generated by users willing to pay a higher monetary rate per bit of traffic injected into the network (dollars per bit) could be given higher priority relative to the traffic generated by the users in the lower end of the rate-per-bit bracket. The goal of the network provider is usually to design algorithms so as to maximize network revenue, which would mean delivering the "promised" QoS. The wired networks can tolerate some degree of network utilization inefficiency in order to satisfy higher priority users (e.g. dropping lower priority packets in order to accommodate higher priority users). In other words, the cost of inefficient network utilization in wired networks is likely to be miniscule when compared to the overall revenue generated.

While this model of priority assignment is very suitable for wired networks, it may not be suitable for resource-constrained systems such as MHWNs. The cost associated with inefficiencies in network utilization is no longer trivial. We argue that in addition to user-assigned priority, the amount of resources "invested" by the network (network investment) into packets or flows should be factored in the process of determining the importance or priority of those respective packets or flows. The amount of investment into a flow could also be used for pricing decisions. To this end, we propose the investment function which is a means of combining and quantifying the various investments (network, user-assigned priority or investment, etc.) made into a packet.

The following example scenarios illustrate the need for an investment function that combines these forms of investment. Assume three descending levels of service quality expectation (and hence relative user investment) for flows: Blue, Red and Yellow. Suppose at a given node we need to drop a packet from a given set of packets because the node buffers are full. Which of the following packets would be the right ones to drop?

1. A Blue packet that has traversed 1 hop or a yellow one that has traversed 5 hops?

2. A Blue packet that has traversed 2 hops under relatively congestion-free conditions or a Red packet that has traversed 2 hops under severely congested conditions, with numerous re-transmissions?

3. A 128-byte Blue packet that has traversed 4 hops or a 1024-byte Red packet that has traversed 6 hops?

From an efficient network utilization perspective, it seems like the Blue packet should be dropped in case 1, but the difference in user investment between Blue service and Yellow service might override that conclusion. Similarly, in the other cases the answer depends on the relative values placed on the different investment factors.

2.2 Investment Function

In this section we introduce the concept of investment function and illustrate its flexibility. Part of this work appeared in [MP06]. We begin by identifying the following dimensions of investment that can be made in network traffic, a list that illustrates the flexibility of the investment function concept, but is by no means exhaustive:

- *Packet Length*: larger packets require larger investment (in both bandwidth, power and buffers) than smaller packets
- *Hops Traversed*: With each successful packet transmission (hop), the cumulative network resources (bandwidth, power, etc.) invested in the packet increases
- *Congestion*: It can be argued that more has been invested in a packet that has been transmitted by a congested node than in one transmitted in a relatively congestion-free environment
- *User Investment*: The customer or user will have invested monetarily in the traffic, with greater relative investment tied to greater service quality expectations.

2.2.1 Sample Investment Function

This diversity of investments in network traffic can be unified by means of an investment function, which can be considered to represent the global value of the packet or packet flow. Here we introduce one possible investment function. Each packet carries in its header a Beginning Investment (I_B) value that is based on the packet size and the relative user investment, and a Network Investment (i_N) factor that reflects number of hops already traversed and network conditions at the upstream nodes. The Current Investment (I_C) value of a packet arriving at a node is computed as follows:

$$I_C = I_B . i_N \tag{2.1}$$

The Current Investment (I_C) is used to make decisions about packet handling, for example, which packets are to be discarded (if necessary) at this node. Details of the investment function computations follow.

The network provider assigns a User Investment (i_U) factor for each service quality level, such that the separation between the i_U values reflect the extent of service differentiation desired. At the source node, the Beginning Investment (I_B) value is computed based on packet size S:

$$I_B = S . i_U \tag{2.2}$$

Also at the source, the Network Investment (i_N) factor is set to some initial value γ where $0 < \gamma \leq 1$ and the Current Investment (I_C) at the source node is computed as:

$$I_C = I_B . \gamma \tag{2.3}$$

At each node (including the source node), after computing and storing the Current Investment (I_C) value of the packet, the Network Investment (i_N) factor is updated:

$$i_N = i_N + (1 - B_A) \tag{2.4}$$

where B_A is the normalized available bandwidth as seen by this node. Finally, embed the Beginning Investment (I_B) value and Network Investment (i_N) factor into the data packet before transmission to the next node. Note that the investment function is generally a time-varying quantity, and its computation constitutes a cross-layer exercise.

Along with the investment function, we introduce a new, related network performance measure: *investment throughput*, defined as investment units delivered to all destinations per unit time. For example, if we use a simpler investment function that is simply hop count times packet size, the investment throughput reduces to *network throughput* expressed in units of hops-bits per second. Specifically, the delivery rate of each flow (in bits per second) would be multiplied by the number of hops traversed by that flow, and the resulting values would be summed over all flows. Investment throughput in this simple example is identical to the *one-hop throughput* in [LBD$^+$01] and similar to the bits-meter per second unit proposed in [GK00], but the concept of investment throughput allows us to generalize the metric. We also introduce an auxiliary metric: *wasted investment rate*, the investment rate of packets that are dropped before reaching their destination. This is calculated in the same manner as investment throughput, but for packets that are dropped.

2.3 Sample Network Objective

The flexibility offered by the investment function can be exploited in various ways. For example, the investment function can be used by packet handling applications to control packet access to node buffers (packet dropping) during congestion, with the goal of minimizing wasted network investment and improving bandwidth parity among flows with different hop counts. Or, the investment function could be used to control access to bandwidth through priority service, control of backoff parameters in wireless network protocols such as 802.11 DCF, etc. Similarly, average flow investment could be used for flow-level control decisions.

We already provided a sampling of flow-related issues in MHWNs in section 1.5. In particular, we focus here on the unfairness exhibited by the MHWN towards longer-hop flows. The problem of poor resource distribution for longer hop flows in MHWNs is well-documented [XS01, Li05, SHS04, RDS+07]. Our aim is to demonstrate the effectiveness of the investment function as a tool to achieve the following sample objectives:

- *improve flow fairness across multiple hops*, and
- *increase network utilization efficiency by reducing wasted investment.*

2.3.1 Sample Service Differentiation Algorithm

In scenarios where flows of variable hop counts and variable user-priorities compete for resources (such as in a MHWN), it seems reasonable that access to the shared resource (bandwidth) should be provided by considering *all* flows or packets that compete for the resource. The contention can emanate from within a given node (*intra-node* contention), or from outside a given node (*inter-node* contention). We will target our algorithm to consider only intra-node contention. Tackling inter-node contention is a complicated task, and is beyond the scope of this proof-of-concept study. We identify it as potential for future work.

To improve flow fairness under intra-node contention, it seems logical that service be accorded to a packet based on accumulated investment of a packet *relative* to other packets currently in the node, rather than strictly user-assigned priority to the packet. By considering accumulated investment, algorithms can be designed to improve multi-hop flow fairness and the efficieny of scarce wireless network resource utilization. We consider a priority-based queuing scheme, with 3 service categories ($3LPQ$) namely *lowest, middle,* and *highest* priority. With conventional priority, one would perform static priority mapping (SPM) under which the priority of a packet does not change as it progresses through the network i.e. the i_U value determines which of the 3 categories the packet of a given flow gets mapped to. In contrast, the investment function allows us to perform dynamic priority mapping (DPM) at each hop, so that treatment given to the current packet is relative to investment carried by packets currently in the node i.e., the current investment value I_C carried in the packet determines the service category that the packet gets mapped to. A relatively larger separation between user-priorities (i_U factors) can be used to approach absolute priority, thus achieving controllable priority.

The DPM scheme used in our study is described as follows: A running mean of the I_C values seen so far (μ_n) is maintained at each node (suffix n denotes n^{th} packet arrival), along with the standard deviation (σ_n). Both are maintained as running variables, updated with the arrival of the I_C value of the nth packet (I_{C-n}).

$$\mu_n = \omega\mu_{n-1} + (1-\omega)I_{C-n} \tag{2.5}$$

$$\sigma_n = \sqrt{\omega.\mu_{n-1}^2 + (1-\omega)(I_{C-n} - \mu_n)^2} \tag{2.6}$$

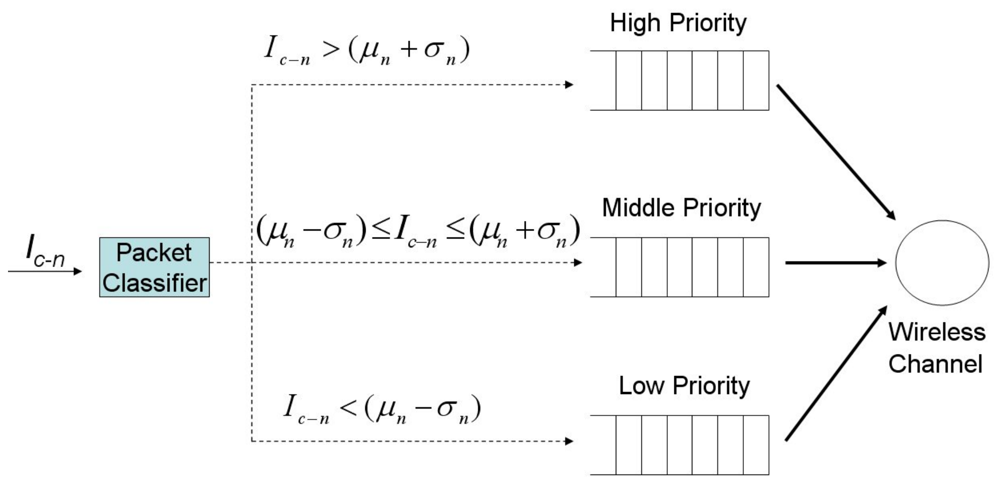
The value for the weight ω is chosen as 0.99 to place a lot of emphasis on the "past" relative to the emphasis on the current investment value. User-defined priorities (when desired) are incorporated into I_C values through i_U values. Dynamic mapping of an incoming packet (n_{th} packet) with current investment I_{C-n}

is performed as follows (see Fig.2.1):

Lowest Priority: $I_{c-n} < \mu_n - \sigma_n$

Middle Priority: $\mu_n - \sigma_n \leq I_{c-n} \leq \mu_n + \sigma_n$

Highest Priority: $I_{c-n} > \mu_n + \sigma_n$



I_{c-n}
Packet Classifier
$I_{c-n} > (\mu_n + \sigma_n)$
High Priority
$(\mu_n - \sigma_n) \leq I_{c-n} \leq (\mu_n + \sigma_n)$
Middle Priority
$I_{c-n} < (\mu_n - \sigma_n)$
Low Priority
Wireless Channel

Figure 2.1. Dynamic Priority Mapping in a 3-Level Priority Queue

2.4 Performance Evaluation

In this section, we use simulations to demonstrate the potential utility of the investment function concept introduced in section 2.2, and illustrate its characteristics. For our simulation studies, we simplified the previous investment function as follows: i_N here is simply the number of hops traversed (regardless of congestion), and packet size S is the same for all packets in a given simulation. We use the sample algorithm introduced in section 2.3.1 to accomplish service differentiation of flows using the investment function.

Keeping in mind the sample objectives from section 2.3.1, we conducted each simulation by choosing one or more options from the following, which illustrate the flexibility of the investment function and its application:

1. *Queuing disciplines:* simple $FIFO$ or 3-level non-preemptive priority ($3LPQ$),

2. *Packet dropping:* Tail-Drop packet dropping (TDD) scheme (drop arriving packets for which there is no buffer space) or Investment-Based Dropping (IBD) scheme, in which the packet to be dropped when a queue would overflow is the packet with the smallest investment value, and

3. *User Priority (UP)*: all equal or differentiated.

Based on the above options, we broadly divide the simulations into two categories:

1. Constant User-investment simulations (CU), and

2. Variable User investment or user-defined priority simulations (VU).

Under CU, I_C is directly proportional to hop count, due to equal packet sizes and equal i_U values across all flows. The baseline case for CU experiment was $FIFO+TDD$ (or simply TDD), and the results were compared against $FIFO+IBD$ (or simply, IBD) and $3LPQ+DPM+IBD$. The results were quite identical between $3LPQ+DPM+IBD$ and $3LPQ+DPM+TDD$. Under VU, the I_C value is directly proportional to the product of hop count and i_U (equal packet sizes). The baseline case in VU is $3LPQ+SPM+TDD$, and the results were compared to $3LPQ+DPM+IBD$. The idea behind VU is to illustrate that the investment function can be used to balance the objective of providing different service types to the users while utilizing resources efficiently and fairly. Separation between i_U values will determine the extent of service differentiation. This flexibility, we believe, is one of the more attractive features of the investment function.

2.4.1 Simulation Scenario

All simulations were conducted using the ns-2 simulator, and both TCP and UDP traffic patterns were used. The transmission range (T_R) of each antenna was approximately 250 meters, while the carrier-sense range (C_R) was approximately 550 meters. We had IEEE 802.11 DCF MAC running on these nodes, with a maximum data rate of 1 Mb/s. AODV was the routing protocol used, while TCP-Tahoe was the flavor of TCP used. To minimize routing overhead, mobility in the nodes was disabled. The total queue size (across all priorities) was fixed to 30 packets in all cases (including $FIFO$). The 802.11 RTS threshold was set to 400 bytes. In all of our VU simulations, the i_U values for lower, middle and highest priorities were set as 1, 2 and 3 respectively. The flows were randomly assigned one of the three i_U values.

We used a grid topology (Fig.2.2) consisting of 16 nodes arranged in a 4x4 grid. Adjacent nodes are separated by 185m (within transmission range), while diagonally opposite nodes are separated by 265m (not within transmission range). The maximum possible hop count in this topology is 6. For each simulation, we chose source-destination (SD) pairs randomly, while enforcing the requirement that there be exactly *12* 1-hop flows, *6* 2-hop flows, *4* 3-hop flows, *3* 4-hop flows, *3* 5-hop flows, and *2* 6-hop flows (total of 30 flows). This was done to ensure that the total network offered load by flows belonging to various hop counts was equal (approximately so for 5-hop flows). To improve the accuracy of our results, we conducted 40 simulation runs for each experiment, with (different) random SD pairs for each run. The duration of each simulation run was 400 seconds.

The performance metrics are *Investment Throughput, Wasted Throughput, Mean end-end flow delay, Mean flow throughput* (TCP only), and *Mean flow*

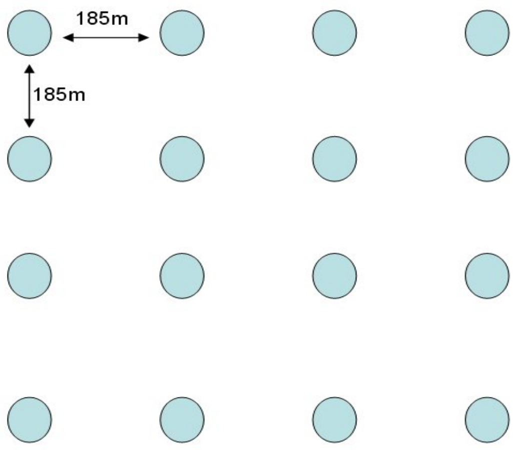


Figure 2.2. 4x4 Grid Topology

packet delivery ratio (UDP only). To assess fairness across multiple hops for the throughput and investment throughput metrics, we make use of *Jain's fairness index* (JFI) [JCH84]. The JFI for a metric X with values $\{x_1, x_2, ..., x_n\}$ is computed as:

$$JFI(x_1, x_2, ..., x_n) = \frac{(\sum_{i=0}^{n} x_i)^2}{n \sum_{i=0}^{n} x_i^2} \tag{2.7}$$

The JFI is bounded in the range $[\frac{1}{n}, 1]$, with higher values indicating higher degree of fairness. For any other of our other performance metrics X, the *variance* of X, in conjunction with an absolute performance measure such as the *mean* of X, is taken as the fairness measure. The mean value is required for correct interpretation of the variance. For example, for identical absolute measures, a reduction in variance implies increased fairness.

2.4.2 UDP Simulations

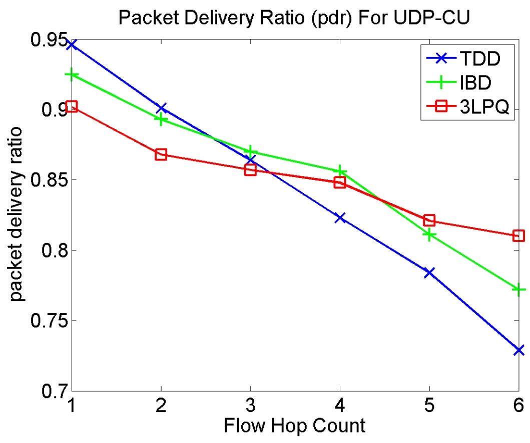
The packet sizes are fixed at 128 bytes (hence the 802.11 RTS/CTS is disabled), and the mean packet inter-arrival time is set as 0.15 seconds to generate a total network load of 204.8 kb/s. Fig. 2.3a and 2.3b show the results for packet delivery ratio (*pdr*) and mean flow delay, respectively, under the various schemes in CU (no user-defined priorities). Please note that there is no SPM under $3LPQ$ in CU; only DPM is performed, which is totally transparent to the user. The $3PLQ + DPM$ scheme was included to illustrate the flexibility provided by the investment function to network providers for packet handling applications, in a manner that is totally transparent to the user.

When compared to the TDD scheme, Figures 2.3a and 2.3b show that the IBD and $3LPQ$ schemes significantly improve *pdr* and delay fairness performance across various flow hop counts. This is summarized in Table 2.1 where, for both *pdr* and delay, the variance across hop count is much smaller using the investment function (IBD, $3LPQ$) compared to conventional TDD, with very little change in mean values. Flows with higher hop counts benefit, while flows with lower hop counts suffer a mild penalty. Hence, one can safely conclude that the investment function indeed improves multi-hop fairness for the $UDP - CU$ scenario.

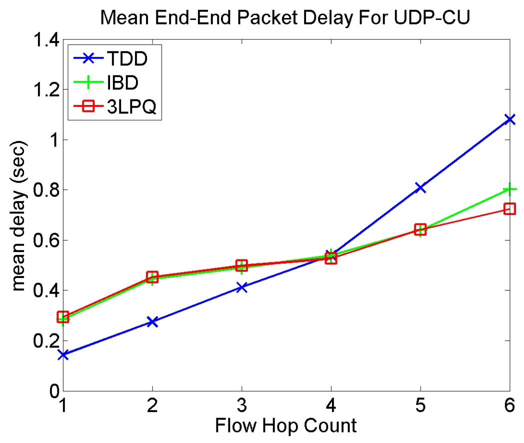
Table 2.1. Mean and Variance of *pdr* and Delay for $UDP - CU$

	Flow Packet Delivery Ratio			Flow Mean Packet Delay		
	TDD	*IBD*	*3LPQ*	*TDD*	*IBD*	*3LPQ*
σ^2	0.0063	0.0031	0.0011	0.1215	0.0311	0.0224
mean	0.8412	0.8545	0.8510	0.5432	0.5333	0.5225

Table 2.2 shows the investment throughput and wasted investment across the three schemes. It can be seen that both IBD and $3LPQ$ marginally improve investment throughput, while simultaneously decreasing wasted investment. From



(a) *pdr*



(b) mean delay

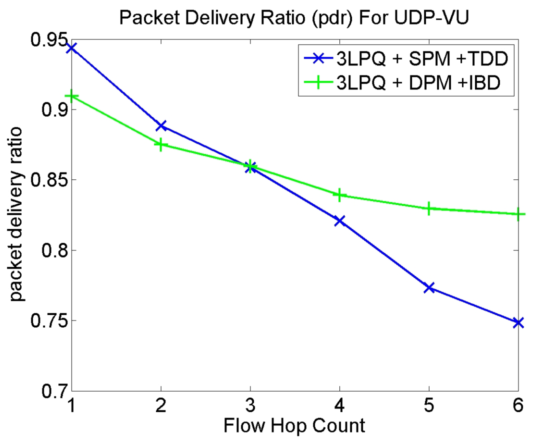
Figure 2.3. *pdr* and mean delay plots for UDP-CU

the high *pdr* values, one can conclude that the network has been loaded just beyond saturation. We achieved larger improvements in network investment throughput at even higher loads, but do not present results here.

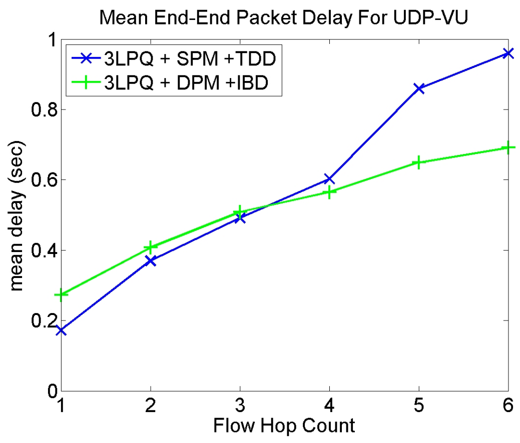
Table 2.2. $UDP-CU$: Network Investment

Investment(Hops-kb/s)	**FIFO + TDD**	**FIFO + IBD**	**3LPQ+ DPM**
Throughput	380.9	387.1	390.7
Wasted	26.6	15.4	12.7

For the VU case (user-defined priorities), we compare $SPM + TDD$ (no investment function) with $DPM + IBD$ (using the investment function). Fig.2.4a and 2.4b compare *pdr* and mean delay performance across flows with different hop counts. Again, a significant improvement in hop-count fairness is evident. Table 2.3 shows the investment throughput and wasted investment across the two schemes for various priorities. The increase in investment throughput is around 5%, while the decrease in wastage is around 50%. Fig.2.5 compares mean delay performance of the two schemes under VU across priorities, showing how the investment function can "soften" the distinction between priorities. This effect is also shown in Table 2.3 (less variance in investment throughput) and Table 2.4, in which DPM has lower variance across priorities. If larger separation is used for user-defined priorities, we expect to see greater distinction between priorities (controllable priorities). Note that increasing the separation between user-investment values (user-priorities) does not impact the SPM scheme at all.



(a) *pdr*



(b) mean delay

Figure 2.4. *pdr* and mean delay plots for UDP-VU

Table 2.3. $UDP - VU$: Network Investment Results

Priority	**Investment Throughput**		**Wasted Investment**	
	SPM	DPM	SPM	DPM
Low	10.37	12.93	1.76	0.49
Middle	12.80	12.69	0.54	0.37
High	13.98	13.36	0.19	0.38
Overall	374.34	391.16	25.09	12.24

As with $UDP - CU$, one also can see from Table 2.4 that the investment function (DPM) is very effective in providing significant improvement in delay and pdr multi-hop fairness, when compared to SPM. The variances of *pdr* and delay values across various hop counts when using DPM are much lower compared to SPM, while offering better mean delay and pdr performance. We conclude that DPM offers better overall performance, while at the same time offering significant improvements in multi-hop flow fairness.

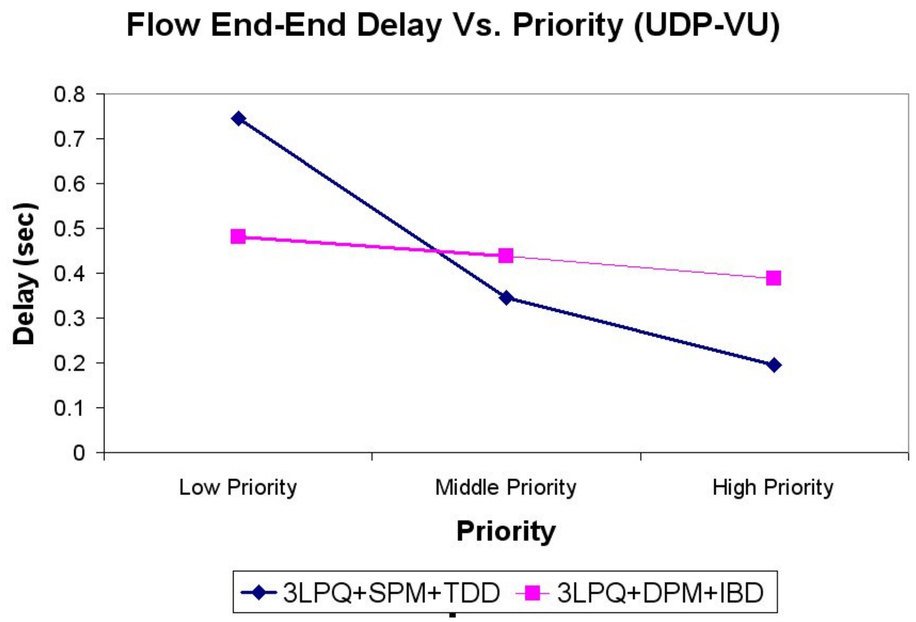


Figure 2.5. $UDP-VU$: Mean Delay Vs. Priority

Table 2.4. $UDP-VU$: Fairness Results

Metric		**SPM**	**DPM**
Pdr vs. Hop count	σ^2	0.0057	0.0009
	mean	0.8333	0.8575
E2e delay vs. Hop count	σ^2	0.0930	0.0236
	mean	0.5700	0.4733
E2e delay vs. Priority	σ^2	0.0882	0.0016
	mean	0.4167	0.4367
Inv. Throughput vs. Priority (JFI)		0.9855	0.9995

2.4.3 TCP Simulations

The packet size for TCP simulation was set to 1024 bytes (RTS/CTS enabled), and the TCP window size was set to 32 kB. Ack packets inherited the investment value of forward packets. Under CU, the total investment throughput values (in hops-kb/s) for TDD, IDB and $3LPQ$ are 910, 886, and 891 respectively, while the corresponding wasted investment values are 6.09, 5.86 and 5.17 respectively. In this case, the investment function has essentially no effect in terms of investment throughput, nor does investment-based dropping substantially reduce the wasted investment, but that wasted investment is small to begin with. We suspect that TCP dynamics (throttling flows that have packet losses, hence minimizing those losses) play a role in keeping wastage to a minimum; however, the fairness properties (throughput and delay) of TCP flows across flows of various hops are greatly improved, as discussed below.

Table 2.5 shows the mean flow throughput across various flow hop counts under CU, and Fig.2.6 shows the mean delay across hop count. For delay, TDD and IBD schemes perform quite similarly, but the $3LPQ$ scheme performs much better by reducing the mean delay of the higher hop counts. However, delay values beyond 3 hops are very high for all schemes because these flows are starved for bandwidth. This asymmetric nature of wireless links and its undesirable interaction with TCP has been studied by A.Rao et al. [RS05]. Our investigation of this problem revealed that the TCP-ACKpackets (reverse direction) at the MAC layer were starved for transmission opportunity due to the *exposed node* problem [BDSZ94], which resulted in poor throughput performance for higher hop count TCP flows.

For illustration, consider the nodes in the first row of the grid topology (let's name them 1, 2, 3 and 4 for convenience). Suppose there are two simultaneous TCP flows originating from node 1 (flow 1: node 1 to node 2; flow 2: node 1 to node 4). When node 1 is transmitting flow 1 packets, under a 802.11 MAC, node 3 does not participate in any transmission or recepition activity because node 3 is within carrier-sense range of node 1. However, node 4 is unaware of the transmission from node 1, and the resultant inability of node 3 to respond. This is the exposed node problem in 802.11 networks. Thus, any attempts from node 4 to send a TCP-ACK to node 3 (final destination to node 1), fails with *high probability*. This is because the rate of flow 1 is much higher than the rate of flow 2 (inherent multi-hop wireless characteristic), and thus the probability of a flow 1 packet occupying the channel is much higher than the probability of a flow 2 (and thus the TCP-ACK of flow 2) packet occupying the channel. This leads to multiple timeouts in the TCP connection of flow 2, eventually shutting down flow 2. This is similar to the phenomenon noted in Xu et. al. in [XS01]. This relates to the *inter-node* contention that we referred to in Section 2.3.1, and our algorithm is not capable of tackling this problem. A MAC layer solution is needed to alleviate this problem, and we identify this as scope for future work.

Table 2.5. Throughput for $TCP - CU$

Flow Hop Count	Flow Throughput (Kb/s)		
	TDD	*IBD*	*3LPQ*
1	65.2	62.8	53.7
2	7.43	9.81	13.8
3	4.15	2.93	6.25
4	0.18	1.58	3.07
5	0.08	0.67	1.04
6	0.69	0.08	0.14

The dynamic mapping scheme reduces the starvation problem to some extent, as evident from flow throughput values in Table 2.5. Though the throughput gains for higher hop-count flows may not seem substantial in an absolute sense, they are still quite substantial in a relative sense (improvement factors: around 2 for 3-hop flows, and around 17 for 4-hop flows). From Table 2.6, it can be seen that both flow throughputs and investment throughput in schemes using the investment function experience moderate (IBD) to significant ($3LPQ$) increases in fairness (JFI) when compared to the TDD scheme. The IBD scheme is expected to only improve network utilization efficiency, and hence we do not see

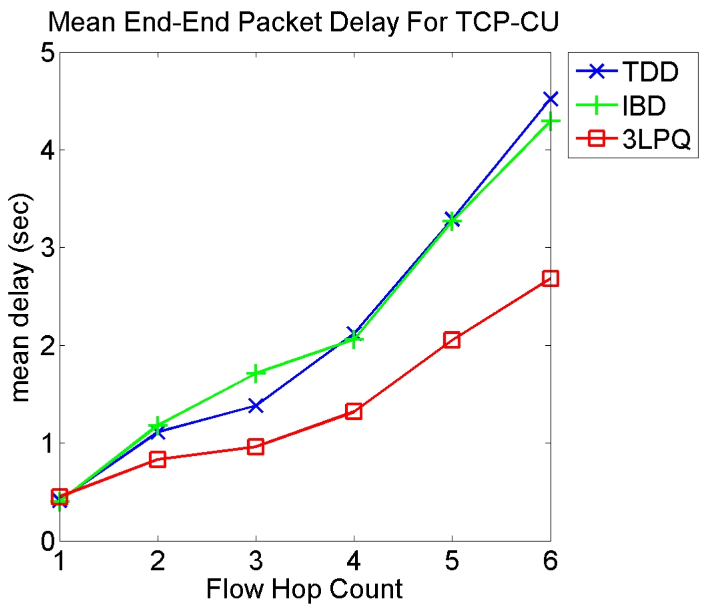


Figure 2.6. TCP: Mean Delay Vs. Hop Count

significant improvements in fairness relative to TDD. For delay, the variance and mean delay values are significantly reduced (almost 70% reduction in variance and 45% reduction in mean delay) in the $3LPQ$ scheme when compared to the TDD scheme. A more sophisticated investment function that captures dynamics of TCP may lead to a higher degree of fairness, and a much better throughput performance.

Table 2.6. $TCP - CU$: Fairness Results

Metric		TDD	IBD	3LPQ
Flow throughput vs. Hops (JFI)		0.2329	0.2495	0.3243
Inv. Throughput vs. Hops (JFI)		0.3430	0.3843	0.5591
Mean delay vs. Hops	σ^2	2.3237	2.0087	0.6903
	mean	2.1383	2.1517	1.3817

For VU simulations, as with UDP, we compared performance of $3LPQ + SPM + TDD$ (no investment function) and $3LPQ + DPM + IBD$ (investment function). The total investment throughput values (in hops-kb/s) for the SPM and DPM schemes are 775.6 and 780.2 respectively, while the corresponding wasted investment values are 7.91 and 6.01 respectively. Again, these gains are modest at best. Table 2.7 shows the throughput performance of both schemes, while Fig.2.7 shows the delay performance under both schemes. Table 2.8 summarizes the hop-count fairness improvements evident from Table 2.7 and Fig. 2.7. Table 2.8 shows a modest improvement in throughput fairness across hop count with DPM compared to SPM, with a more significant improvement in delay fairness. As seen from Figures 2.8a and 2.8b and Table 2.8, DPM again shows significant softening of the distinction between priorities. As with UDP, we argue that the degree of service differentiation under DPM can be controlled by appropriate choice of i_U values.

Table 2.7. Throughput for $TCP - VU$

Flow Hop Count	Flow Throughput (Kb/s)	
	SPM	*DPM*
1	44.99	37.66
2	11.83	16.04
3	9.87	8.73
4	0.98	2.01
5	0.24	2.19
6	0.06	0.36

Table 2.8. $TCP - VU$: Fairness Results

Metric		**SPM**	**DPM**
Flow throughput vs. Hops (JFI)		0.3403	0.4248
Inv. Throughput vs. Hops (JFI)		0.3403	0.4248
Mean delay vs. Hops	σ^2	3.1730	0.4252
	mean	2.2217	1.1517
Inv. Throughput vs. Priority (JFI)		0.5880	0.9863
Mean delay vs. Priority	σ^2	0.0336	0.0013
	mean	0.6410	0.6002

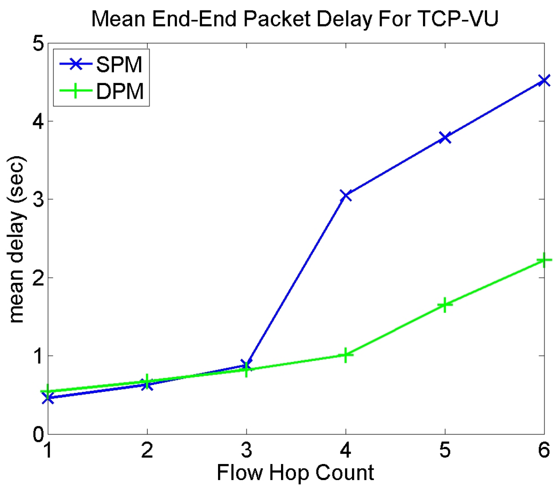
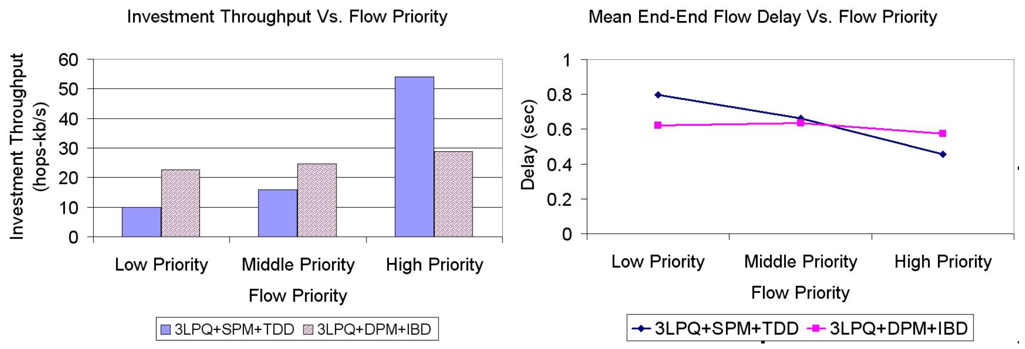


Figure 2.7. TCP: Mean Delay Vs. Hop Count



(a) invst. throughput (b) mean delay

Figure 2.8. $TCP - VU$: Priority Results

2.5 Optimality of Results

While our *heuristics-based* definition of the investment function and service differentiation algorithm seemed to achieve the desired objectives, the improvements are probably far from optimal. In fact, we do not even know the “extent” of the sub-optimality of our results. Only if we know the “optimal” values of our network objectives, we can design algorithms to perform in a manner so as to achieve optimal or near-optimal network objectives. Frequently, such network design problems are cast as flow-optimization problems to obtain optimal design parameters. The theory and applications of network flow optimization is very rich in wired networks. Sophisticated algorithms and techniques have been developed to solve specific flow problems in wired networks (e.g. Ford-Fulkerson algorthim [LD56] to solve $MAXFLOW$). Unfortunately, network flow problems arising in MHWNs are difficult to solve by direct application or extension of techniques from the wired domain due to differences in the concept of a link in wired vs. wireless networks. In both domains, each link represents a transmission pipe to the neighbor it is connected to, and the pipe's capacity is constrained and known *a priori*. In wired networks, the residual capacity (available bandwidth) of the link is independent of the neighbor transmissions due to the dedicated nature of the pipe. On the other hand, in wireless networks, since nodes use a shared medium for transmission, the link is no longer dedicated; it can be viewed as a transmission pipe with variable residual capacity that depends on neighbor nodes' transmissions, i.e., interference. Thus, a constrained optimization approach is attractive in these cases.

In a constrained optimization approach, an objective function that needs to be optimized is specified, followed by a list of constraints under which an optimal solution needs to be found. The optimization can either be a Linear Programming (LP) or Non-Linear Programming (NLP) problem based on the objective function and the constraints of the problem. The constraints consists of both *problem-specific* constraints, and *network-imposed* constraints. The problem-specific constraints are related to the problem at hand that needs to be optimized, and could vary based on the specific problem. On the other hand, the network-imposed constraints are those that are imposed by the specific network for which the flow optimization is being done. The network-imposed constraints are fundamental and inherent to the given network, and do not usually change for multiple optimization problems formulated for the given network.

Formulating the practical problem at hand into an optimization problem is an art in itself. Formulating an equivalent optimization problem for the dual-objective problem from section 2.3 is quite challenging and requires a lot of effort. In this dissertation, we do not strive for such a formulation; instead we focus on a particular difficulty associated with solving *all* flow problems in MHWNs, and develop techniques to solve it. More specifically, we focus on the problem of formulating the network-imposed constraints for MHWNs. Through an example, we will show that the network-imposed constraints are rather trivial to be formulated for the wired networks, while surprisingly difficult for MHWNs. The difficulty is due to the adjacent-node interference problem caused by the shared nature of the wireless medium.

Consider the following motivating example: we consider the simple problem of finding the maximum flow between a given source-destination in a given network ($MAXFLOW$). We will first formulate the problem for a wired network scenario (formulation adapted from [Lue84]), followed by a wireless networking (MHWN) scenario, illustrating the key differences and difficulties in the process.

Given n nodes in a multi-hop *wired* network, we wish to find the maximal value of flow rate f between a given source (node r) and destination (node s) in the network. The flow on the link uv is given as x_{uv}. The specific nature of the problem imposes the *flow-conservation* constraint (problem-specific constraint) that each node must obey: the net flow *outflow* from the source be exactly f, and the net flow *inflow* into the destination be exactly f, while the net flow into (from) the other nodes be exactly zero. The network configuration imposes the *link capacity constraints* (network-specific constraint) that the flow on each transmission link x_{uv} cannot exceed the total link capacity (rate) c_{uv}. Note that $c_{uv} = 0$ if nodes u and v do not have a link between them.

Maximize $\quad f$

Subject to

1. *Flow balance equations*

$$\begin{aligned}
\sum_{\substack{j=1\\ j\neq r}}^{n} x_{rj} - \sum_{\substack{j=1\\ j\neq r}}^{n} x_{jr} - f &= 0 \quad \textit{(source)}\\
\sum_{\substack{j=1\\ j\neq i}}^{n} x_{ij} - \sum_{\substack{j=1\\ j\neq i}}^{n} x_{ji} &= 0 \quad i = 1, 2, .., n; i \neq r, s\\
\sum_{\substack{j=1\\ j\neq s}}^{n} x_{sj} - \sum_{\substack{j=1\\ j\neq s}}^{n} x_{js} + f &= 0 \quad \textit{(sink)}
\end{aligned}$$

2. *Capacity constraints*

$0 \leq x_{ij} \leq c_{ij}, \quad \forall i, j = 1, 2, .., n$

Figure 2.9. LP formulation for MAXFLOW between source r and destination s in a wired network

The $MAXFLOW$ formulation for a MHWN is quite similar to the formulation for wired networks shown in Fig.2.9, with the exception of the capacity constraints. Links between various nodes no longer have "exclusive" capacities; instead the total channel capacity (rate) is shared among multiple nodes (transceivers). In particular, for any node u, if C is the capacity (maximum data rate) of the channel used by the transceiver of u, then the following capacity constraint must be satisfied:

$$\sum_{\forall v \in N_u^C} x_{uv} + \sum_{\forall v \in N_u^C} x_{vu} + \xi_u \leq C, \quad 0 \leq x_{uv}, \xi_u \leq C \tag{2.8}$$

Here N_u^C is the set of "communicating neighbors" of u. All nodes that lie within the transmission range of a given node are called the *communicating neighbors* of that node (nodes within T_R of u), since a node can have communication links only with nodes that are within its transmission range. The first sum is the flow out of node u and the second is the flow into node u; these are considered separately due to the shared medium for transmit and receive. The value ξ_u is the effective "interference" sensed by node u. The quantity ξ_u arises because of the shared nature of the wireless medium, and this is precisely what we have been referring to as adjacent-node interference. For analytical purposes, first we need to build an "interference model" that captures the interference relationship between the nodes. Under a given interference model, ξ_u depends on a variety of factors, chief of them being the actual transmission rates of the nodes in the network (x_{uv}) and the scheduling algorithm used to provide channel access to the various nodes. The scheduling algorithm for a given interference model also determines whether ξ_u can be expressed in polynomial form using the x_{uv} values. In the next chapter, we develop the basic tools necessary to formulate the capacity constraints in MHWNs.

2.6 Summary

One of the major contributions of this dissertation is the introduction of a new concept in networking called the investment function, the development of which will, we believe, aid in more effective service differentiation procedures to tackle the existing problems in MHWNs. Through a proof-of-concept study, we formulated a specific definition of the investment function, in addition to a simple, customized service differentiation algorithm to achieve sample network objective of improving multi-hop fairness in multi-hop wireless networks, in addition to improving network utilization efficiency. Through our simulations, we demonstrated two of the many ways that the investment function can be used for better network performance. Our TCP and UDP simulation results with a grid topology indicate that the investment function can provide substantial improvement in throughput and delay fairness properties across multiple hops in addition to substantial reduction in network wastage for UDP flows. We have also argued how the investment-based approach can provide controllable-differentiation priority.

We noted that our results could be far from optimal, and to design optimal algorithms, we need to obtain optimal design parameters. Often, these kinds of problems are solved by first formulating these problems as *network flow optimization* or simply network flow problems, and then solving them using well-known optimization techniques. Unfortunately, solving flow problems in MHWNs is very hard due to the complexity in modeling and computing the adjacent-node interference value. The rest of this dissertation is focused on studying and developing techniques and methods to solve network flow problems in MHWNs.

Chapter 3

Capacity Constraints in MHWNs

3.1 Interference Modeling in MHWNs

To formulate the capacity constraints, we need to model and compute interference. To this end, we first represent the MHWN using a graph model. Then, under the assumption of a specific scheduling algorithm, we compute the interference using graph theoretic tools, and then formulate the capacity constraints. Assume that we are given a MHWN of n nodes $(1, 2, .., n)$, and their respective position coordinates in two dimensional space. Let $d(u, v)$ represent the euclidean distance between any two nodes u and v. We make the following assumptions while constructing our mathematical model that represents the interference and connectivity relationships between the nodes in the network:

1. The nodes are considered to be stationary
2. All nodes use the same frequency and bandwidth for transmission (hence use TDMA for channel access)
3. Presence of bi-directional wireless links between nodes that are neighbors

4. Homogenous antenna properties *i.e.*, all node antennas have same T_R and C_R values

5. Fluid-flow model for node transmissions

6. Perfect (collision-free) scheduling for node transmissions

7. *Transmitter (Tx)* model of interference (described below)

The Tx-model was first introduced in [YPK03] to analyze the capacity of random ad hoc networks. In the Tx-model, a transmission from a node u is succesfully received by all of its neighbors within a distance of T_R if and only if for any other transmitter v, $d(u, v) > (1+\delta)(T_R+C_R)$, where δ is the guard band. In our work, we shall assume that $\delta = 0$. Under the Tx-model, two nodes that lie within a distance of $T_R + C_R$ from one another can be considered to "interfere" with each other's transmission.

The intuition behind this model is that one could consider two transmitting nodes u and v as interfering with one another if the transmission of one node interferes *directly* with the other (nodes within a distance of C_R of one another) or *indirectly* (one or more receivers of one node within C_R of the other node). Thus, we coin the term *interference range* ($I_R = T_R+C_R$) to denote the maximum distance up to which a transmitting node's "impact" can be felt. In other words, it is the distance of separation between two nodes that will guarantee that two nodes can transmit simultaneously regardless of placement of receivers. All nodes that lie within interference range of a given node are *interfering neighbors* of that node.

It must be noted that the Tx-model is quite a conservative model. The Tx-model would identify two nodes u and v as interfering with one another if

$C_R \leq d(u,v) \leq I_R$, even if u and v do not have any receivers (within T_R of the transmitting nodes) located in the region of intersection of circles representing their respective interference ranges (common interference region). Moreover, the Tx-model would consider simultaneous transmissions by u and v as interfering with one another even if respective receivers are located outside the common interference region. Figure 3.1 gives an example which shows the conservative property of the Tx-model. In this example, four nodes A, B, C, and D are considered, with $C_R \leq d(A,B) \leq I_R$, and $d(A,D), d(B,C) < T_R$. Based on our definitions of transmission range and carrier-sense range, the transmission from A to D will not interfere with the transmission from B to C, and can occur simultaneously. However, the Tx-model would identify A and B as interfering with one another, and prohibit these nodes from transmitting simultaneously.

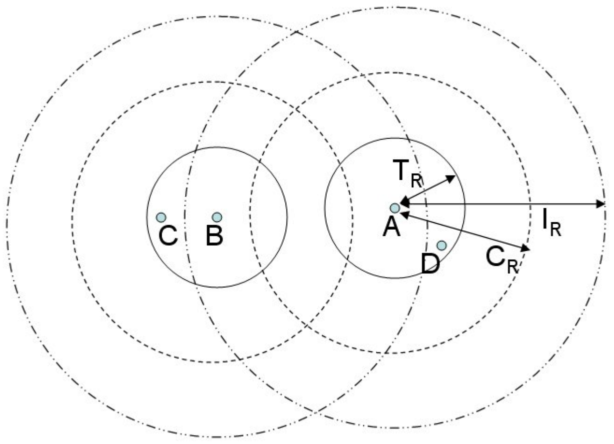


Figure 3.1. Example showing the conservative nature of the Tx-model of interference

A more realistic model is the protocol model, which was introduced in [GK00]. Under the protocol model, if node u transmits to node v, the following two conditions have to be satisfied for successful transmission:

1. $d(u,v) \leq T_R$

2. No other node w is transmitting simultaneously such that $d(w, v) \leq C_R$

Though the protocol model is more realistic than the Tx-model, the Tx-model has its own merits. The Tx-model is a node-level interference model, whereas the protocol model is a link-level interference model. That is, in the Tx-model, the interference can be considered to occur between two *nodes* themselves, as opposed to the protocol model where the interference needs to be considererd as occuring between two *links*. This factor becomes important in the context of graph-theoretic modeling of interference, where the size of the graph depends on the number of interfering entities. As the number of links in a network of n nodes is $O(n^2)$, a node-level model requires a significantly less computational resources when compared to a link-level model. Moreover, as we will see later, the node-level model enables us to use a special type of graph called the *Unit Disk Graph (UDG)* for modeling the interference, which has certain advantages when compared to a general graph model (details later).

As one can see, both the the Tx-model and the protocol models are rather simplistic (binary on-off interference, based on distance). The most accurate model for communication would be the *Physical Model*, also introduced in [GK00]. Under the physical model, for the transmission between u and v to be successful, the signal-to-noise ratio measured at v due to the transmission from u (SNR_{uv}) should exceed some threshold SNR_{th} (already discussed in section 1.2). Though the physical model is more realistic than the Tx-model, we use the Tx-model because it facilitates analytical tractability when using graph-theoretic modeling.

The TDMA mode of channel access was assumed only for ease of exposition. Under TDMA mode of access, the resource under contention is time-slots. The concepts we develop in this dissertation can be extended to other modes of channel access, where interference will be defined appropriately for the resource under contention. For example, under FDMA, the various carrier frequencies used for modulation are the resources under contention.

3.2 Unit Disk Graph Model

We use the Unit Disk Graph (UDG) model to represent connectivity and interference relatanship between the nodes. In a UDG $G(V, E)$ with vertex set V and edge set E, there is an edge uv between vertices u and v if and only if (*iff*) the Euclidean distance between u and v , $d(u, v)$, is less than or equal to 1:

$$E = \{uv | d(u, v) \leq 1 \, \forall u, v \in V\}$$

From this definition, the UDG is a natural choice to represent the communication and interference relationship between the nodes in a MHWN.

To capture the relationship between communicating neighbors of a graph, we define the *Communication UDG*, $G(V, E^C)$, wherein all inter-nodal distances are normalized with respect to T_R. The vertex set V corresponds to the nodes in the MHWN, and the edge set E^C corresponds to the links between communicating neighbors.

$$E^C = \{uv | d(u, v) \leq T_R, \forall u, v \in V\}$$

Similarly, we define the *Interference UDG*, $G(V, E^I)$, wherein all inter-nodal distances are normalized with respect to I_R. The vertex set V corresponds to the nodes in the MHWN, and the edge set E^I corresponds to the links between interfering neighbors. Under the Tx-model, if two nodes in the interference UDG share an edge (adjacent nodes), then they cannot transmit simultaneously.

$$E^I = \{uv|d(u,v) \leq I_R, \forall u, v \in V\}$$

The notion of bidirectional links is implicit in the definition of these UDGs i.e., if node u can communicate (interfere) with node v, then it also means that node v can communicate (interfere) with node u.

For a given node $u \in V$, we can define the set of communicating neighbors (N_u^C) and interfering neighbors (N_u^I) as follows:

$$N_u^C = \{v|d(u,v) \leq T_R, \forall v \in V, v \neq u\}$$

$$N_u^I = \{v|d(u,v) \leq I_R, \forall v \in V, v \neq u\}$$

The communication UDG can be used for several purposes in the MHWN, such as determining connectivity, routing decisions, etc. The interference UDG can be used for computing the interference sensed by the nodes in the MHWN. As an example, consider a 6-node chain topology as shown in Fig.3.2. With $T_R = 250m$ and $C_R = 250m$ $(I_R = 500m)$, the corresponding communication and interference UDGs are shown in Fig.3.3 and Fig.3.4, respectively. Table 3.1 shows the set of communicating and interfering neighbors for each node.

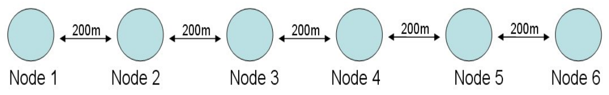


Figure 3.2. Chain Topology of 6 nodes. Inter-node distance $(d(u,v) = 200m)$

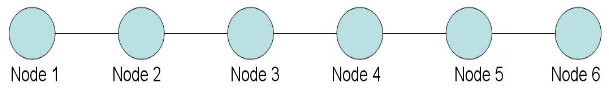


Figure 3.3. Communication UDG for 6-node chain topology. $T_R = 250m$

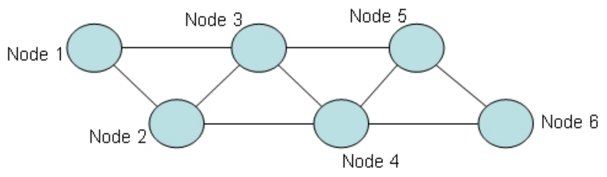


Figure 3.4. Interference UDG for 6-node chain topology. $I_R = 500m$.

Table 3.1. Communication neighbors and Interfering neighbors for 6-node chain topology

Node	*Communicating Neighbors* (N^C)	*Interfering Neighbors* (N^I)
1	2	2, 3
2	1, 3	1, 3, 4
3	2, 4	1, 2, 4, 5
4	3, 5	2, 3, 5, 6
5	4, 6	3, 4, 6
6	5	4, 5

3.2.1 Limitations of UDG model

The UDG model is a *node-level* model, and very suitable for modeling the Tx-model of communication. In some situations, it may be required to capture the relationship between the *links* of neighboring nodes. Such situations may arise if one is interested in analyzing a certain channel access protocol (e.g. 802.11), where links may be in conflict due to protocol behavior, though the actual nodes may not be (e.g. RTS-CTS handshake in 802.11). We accept this limitation in our current work, but plan to address it in future work.

3.3 Scheduling and Interference Computation

In this subsection, we will look at some concepts that relate scheduling and the interference experienced by the nodes in the network. We first define the rate vector $X = [\{x_{uv}\}]$, where x_{uv} is the total transmission rate from node u to node v,

$\forall uv \in E^C$. To capture the traffic rate information by each node, we augment the interference UDG $G(V, E^I)$ to a *weighted* interference UDG $G(V, E^I, W)$, where $W = [w_1, w_2, .., w_n]$ is the vector of node weights. The node weight w_u for a given node u is computed by summing the total outgoing link rates of the node:

$$w_u = \sum_{\forall v \in N_u^C} x_{uv}$$

For ease of exposition, we normalize the maximum channel data rate (wireless bandwidth or capacity) achievable by the transceiver of each node to 1. Under a time-slotted system, we can interpret each link (node) rate as the fraction of time slots alloted for transmission for that particular link (node), in a given *super-frame* or time-slot window of length γ time slots, where γ can be thought of as the number of timeslots in one second for the given maximum channel data rate. Given a rate vector X, we assume that there exists a *scheduling* algorithm that computes a global schedule S which contains information about the nodes that are permitted to transmit in any given time slot. The schedule S has the following properties:

- each node is alloted transmission timeslots such that no two interfering nodes transmit in the same timeslot
- the schedule is M slots in length (also called the *span* of the schedule)
- the fraction of time slots alloted to a given node in the span of M slots is at least as high as the respective rate requirements of that node

A schedule S is *feasible* if and only if $M \leq \gamma$. Given a feasible schedule S, the same transmission order is repeated periodically, with a period of γ slots.

From the Tx-model of interference, two nodes *interfere* with one another if they cannot transmit simultaneously. Since we assume a CSMA/CA model of channel access, a given node cannot receive and transmit simultaneously. Thus the total interference sensed by a given node u, represented as e_u, may also of consist traffic intended for u (i.e. in reference to (2.8) $e_u = \sum_{\forall v \in N_u^C} x_{vu} + \xi_u$). For a time-slotted system, the interference value $e'_u = \gamma.e_u$, expressed as the number of interfering time slots, is computed by counting the number of time slots in S in which at least one interfering neighbor of u is scheduled. If $I(u,i)$ is an indicator variable which has a value of 1 if any node $v \in N_u^I$ is scheduled in time-slot i of S (0 otherwise), then:

$$e'_u = \sum_{i=1}^{M} I(u,i) \tag{3.1}$$

Thus, the extent of interference experienced by a given node depends on the following:

1. network topology
2. actual traffic generated (transmission rate vector X) by the interfering neighbors
3. span M of the schedule, which depends on the scheduling algorithm used

The span M of the schedule (and thus the interference values) depends on the scheduling algorithm used. An optimal scheduling scheme returns the minimal value of M, leading to the minimal value of interference as experienced by a given node. This could potentially increase per node throughput, thus maximizing network throughput performance. It can be argued that an assumption of optimal scheduling is essential to obtain benchmark or upper bound value on network performance. It becomes more compelling to assume optimal scheduling since we are focused on developing an optimization framework for MHWNs. Thus, in this dissertation, we only consider optimal scheduling, though we provide an illustration (later in this chapter) with a pessimistic scheduling algorithm as well.

Let us define the following *capacity constraints*:

$$\forall u \in V : \sum_{\forall v \in N_u^C} x_{uv} + e_u \quad \leq \quad 1, \quad 0 \leq x_{uv}, e_u \leq 1 \tag{3.2}$$

$$\forall u \in V : \sum_{\forall v \in N_u^C} x'_{uv} + e'_u \quad \leq \quad \gamma, \quad 0 \leq x'_{uv}, e'_u \leq \gamma \tag{3.3}$$

Two equivalent representations of the capacity constraints are given. The set of capacity constraints in (3.2) are described in units of normalized bits per second, while the set of constraints in (3.3) are given in units of time slots, where $x'_{uv} = \gamma.x_{uv}$, $e'_u = \gamma.e_u$, $w'_u = \gamma.w_u$, and $W' = \gamma.W$.

For a given rate vector X, the knowledge of the span M of the schedule is sufficient to assess feasibility of X. We will show that, equivalently, the knowledge of the interference values (e_u) is sufficient for feasibility assessment. In particular, we will show that the capacity constraints are both necessary and sufficient conditions to assess feasibility of X. Such feasibility assessments of some rate vector X frequently arise in flow-related optimization problems in the MHWN domain, and the capacity constraints are critical tools in that aspect. These capacity constraints are network-imposed constraints, and any rate vector X in flow-related optimization in MHWNs must satisfy the network-imposed constraints, in addition to satisfying specific constraints imposed by the problem definition. Hence our interest in the capacity constraint formulation.

Lemma 1. *For a given MHWN, a given rate vector X is schedulable or feasible under a given scheduling algorithm if and only if $\forall u \in V$ in the network, X satisfies the capacity constraints.*

Proof: Let us start by assuming that the scheduling algorithm provides a global schedule S of length M time slots. Let us prove the sufficiency conditions by contradiction. We assume that X satisfies (3.3), but is infeasible ($M > \gamma$). Out of the total n nodes in the network, let us assume that exactly k nodes are scheduled in timeslots $[\gamma + 1,\ M]$. Without loss of generality, we number the k nodes as $1, 2, .., k$. We consider node 1 and assume that it has been scheduled to transmit in exactly p slots ($0 < p \leq M - \gamma$) in $[\gamma + 1,\ M]$. This implies that *each* of the timeslots in $[1,\ \gamma]$ either has node 1 scheduled, or some interfering neighbor scheduled (otherwise, some of the p transmissions could have been scheduled in $[1,\gamma]$). Using a similar approach from (3.1) we compute the interference e''_1 for node 1 in $[1,\gamma]$ (note: $e''_1 \leq e'_1$). Then:

$$\begin{aligned} \sum_{\forall v \in N_1^C} x'_{1v} - p + e''_1 &= \gamma \\ => \sum_{\forall v \in N_1^C} x'_{1v} - p + (e'_1 - \delta e) &= \gamma, \quad \text{where } \delta e = e'_1 - e''_1 \geq 0 \\ => \sum_{\forall v \in N_1^C} x'_{1v} + e'_1 - (p + \delta e) &= \gamma \qquad (3.4) \end{aligned}$$

Since $(p + \delta e) > 0$, if (3.4) has to be satisfied, then it implies that $\sum_{\forall v \in N_1^C} x'_{1v} + e'_1 > \gamma$. This contradicts our initial assumption that X satisfies the capacity constraints (3.3) for all nodes, including 1. A similar argument can be made for the remaining nodes $2, 3, .., k$ and hence the proof.

Now we prove the nececssary condition that a rate vector X must satisfy the capacity constraints in order to be feasible or schedulable under a given scheduling algorithm. We adopt a proof-by-contradiction strategy, and assume that for a schedulable rate vector X, there is at least one node $z \in V$ that violates the capacity constraint. The constraint for z can be listed as $\sum_{\forall v \in N_z^C} x'_{zv} + e'_z = M', M' > \gamma$. It follows that $\max_{\forall u \in V} \sum_{\forall v \in N_u^C} x'_{uv} + e'_u \geq M'$. Since the scheduling algorithm returns a schedule of length M, it must be that $M' \leq M$. It follows that that $M > \gamma$, thus contradicting the feasibility of X, and hence the proof. $\square$

To illustrate computation of interference, we will consider the same 6-node chain topology shown in Fig.3.2, augmented with some traffic rates generated by each node. More specifically, we will assume that each node generates traffic at the rate of 1 unit per second on each of its outgoing links, as shown in Fig.3.5. The weighted interference UDG corresponding to the 6-node chain topology with traffic rates (Fig.3.5) is shown in Fig.3.6. Nodes 1 and 6 each generate traffic at the rate of 1 unit per second (and thus a node weight of 1), while the remaining nodes generate traffic at the rate of 2 units per second (node weight of 2). For convenience in the remainder of the dissertation, we drop the word "weighted" when referring to the weighted interference UDG, and instead simply refer to it as an interference UDG.

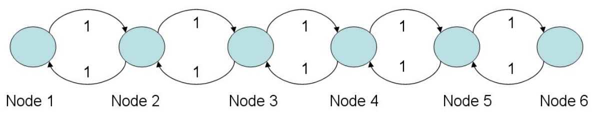


Figure 3.5. 6-node chain topology augmented with traffic rates. Each node transmits at the rate of 1 unit per second on every outgoing link

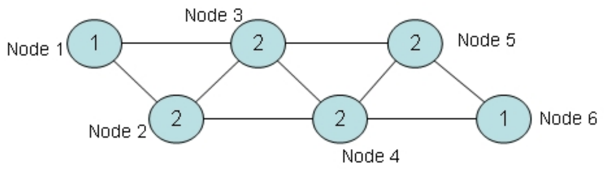


Figure 3.6. Weighted Interference UDG for 6-node chain topology. Weights are enclosed within circles. A node's weight corresponds to it's total outgoing transmission rate.

Without loss of generality, we assume that the transmission of 1 unit of data requires exactly 1 time slot. First, we look at how these transmissions can be scheduled in a feasible manner. Since the total number of transmission units for the 6-node chain topology is 10, a pessimistic scheduler could schedule all the 10 transmissions in 10 non-overlapping timeslots, as shown in Fig.3.7. This is wasteful, as this schedule has been arrived at without any consideration for spatial reuse of the carrier frequency. As the interference UDG would show, some nodes can be scheduled simultaneously. For e.g., the pairs of nodes (1,4), (2,5) and (3,6) can be scheduled to transmit in the same timeslot. The number of timeslots required for realizing a feasible schedule depends on the scheduling algorithm used. For the 6-node chain topology under consideration, an optimal schedule requires 6 timeslots, as shown in Fig.3.7. The interference values defined in (3.1) can be computed by looking at the global schedules in Fig.3.7. For example, for node 4, the interfering neighbors are nodes *2, 3, 5* and *6*, generating a total of 7 units of traffic, which can be scheduled in 4 timeslots (slots 2 - 5) under optimal scheduling. This results in an interference rate of 4 timeslots per superframe, as sensed by node 4. Under pessimistic scheduling, the interference rate can be computed as 7 timeslots.

So far, we have assumed that given a rate vector X, there existed a method or algorithm to compute the optimal schedule. We then compute interference values, and test X for feasibility using the capacity constraints. We will now focus our attention towards formal methods that can accomplish the optimal scheduling process, and provide us with the value of M. We emphasize that we are only interested in the interference values in the context of feasibility of a given rate vector X under some scheduling algorithm, and not in the values themselves in isolation. Readers famililar with graph coloring may recognize that the optimal scheduling procedure shown in Fig.3.7 can be cast as an *optimal* weighted graph coloring problem. Hence, a discussion of graph coloring is in order.

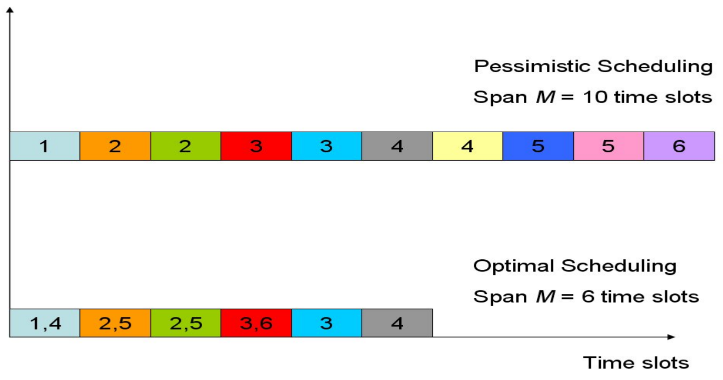


Figure 3.7. Pessimistic and Optimal scheduling. The scheduled node indices are shown within rectangles

3.4 Graph Coloring and Interference

Graph coloring is the process of assigning colors to the vertices of an unweighted graph G (one color per vertex), such that no two adjacent vertices share the same color. If the graph is weighted (integer weights only), then the number of colors required to color a vertex equals the weight of that vertex, and the coloring procedure is referred to as *weighted coloring*. Weighted coloring on a weighted graph $G(V, E, W)$ is usually accomplished by first transforming G into an unweighted graph $H(V^H, E^H)$, and then performing an unweighted coloring

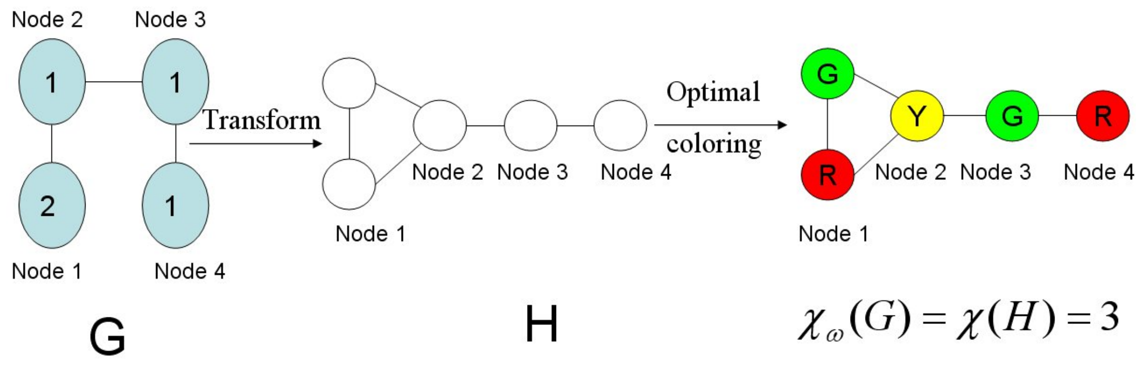


Figure 3.8. For a sample UDG (G), transformation from a weighted graph to unweighted graph (H), followed by optimal coloring. Three colors are used: *Red* (R), *Green* (G) and *Yellow* (Y)

procedure on H. The transformed graph H is constructed as follows: for each vertex $u \in V$ in G with an integer weight of w_u :

1. replace u by K_{w_u}, where K_{w_u} refers to a *clique* (a complete subgraph) of size w_u, and

2. connect (using edges) every vertex of K_{w_u} with every vertex of $K_{w_v}, \forall v|(u,v) \in E$.

If the coloring is done using an optimal (minimum) number of colors, then the coloring procedure is called *Optimal coloring.* The optimal number of colors required to color a graph is called the *chromatic number*, χ of the graph. For a weighted graph G, the chromatic sum or weighted chromatic number χ_w of G equals the chromatic number of the transformed unweighted graph H i.e.,

$$\chi_w(G) = \chi(H)$$

Figure 3.8 shows the transformation and optimal coloring of a sample UDG.

Graph coloring is frequently used for resource allocation and scheduling problems. In multi-hop radio and cellular networks, graph coloring has been extensively used for frequency assignment. For examples, see [Hal80, Lei79, Wer85, SDS97]. First, the competing entities and their relationship is represented as a graph, and coloring is performed on this graph to arrive at a resource allocation decision. In our case, a time-slot is the resource under contention, and the contention relationship between the competing entities (node transmission attempts) are represented using the interference UDG.

All vertices that are colored using the same color are said to belong to a *coloring class* i.e., if $c(v)$ is the color of vertex v, then set of all coloring classes can be represented as $V_i = \{v | c(v) = i, \forall v \in V\}$. The transmission scheduling problem discussed in the previous section can be solved by first transforming the weighted graph to an unweighted one, where the vertices in the transformed graph correspond to individual node transmission units, followed by optimal coloring of the transformed graph. Optimal scheduling is accomplished by allowing all vertices colored using the same color to transmit in the same time slot. The total number of colors required to color the graph represents the span M of the schedule. In optimal scheduling, there are exactly $\chi_w(G)$ coloring classes, and the nodes in each coloring class can be scheduled simultaneously. Thus, the set $\{V_i\}$ gives *an* optimal schedule of transmissions. In the optimal schedule in Fig .3.7, the set of coloring classes can be given as { {1, 4}, {2, 5}, {2, 5}, {3, 6}, {3}, {4} }.

Unfortunately, for most graphs (including UDGs), computing the chromatic number is an NP-complete problem [BCD90]. No known polynomial time algorithms exist for optimal coloring, except for few specific classes of graphs. Thus, computing M is a potentially expensive task. Hence, we need to look at alternate methods of computing the optimal schedule. One popular solution is to bound the chromatic number using easily computable quantities. Plenty of polynomial-time approximate coloring algorithms exist, which produce upper bounds on the chromatic number. An approximate coloring can be thought of as producing a schedule of span ηM for some $\eta \geq 1$. A rate vector X is then feasible if $\eta.M \leq \gamma$. For a complete survey on approximate coloring algorithms in UDGs, see [EF06]. In the next section, we will describe the use of a well-known graph invariant, the *clique number*, as a bound on the chromatic number, and through an empirical study, we observe that the bound is quite tight (η close to 1.0) for practical scenarios.

3.5 Cliques and Clique Number

A clique is a complete (fully interconnected) sub-graph in a given graph G, and a maximal clique is a clique that is not contained in any larger clique. The maximum clique number (the modifier "maximum" is often omitted) ω(G) is the maximum size of all maximal cliques. For weighted graphs, a *Maximum Weighted Clique (MWC)* is a maximal clique with maximum sum of vertex weights. The *maximum weighted clique number* ω_w is the sum of vertex weights of the MWC. Let Q denote the set of nodes that forms a maximal clique, and let $|Q|$ denote the cardinality (number of elements) of the set Q. Let us suppose that in a given weighted graph G, there are m maximal cliques enumerated as $\{Q_1, Q_2, ..., Q_m\}$, and the subgraph induced by the nodes of MWC be denoted as G^M.

$$\begin{aligned} \omega(G) &= \max_i(|Q_i|), i = 1, 2, .., m \\ \omega_w(G) &= \max_i \left(\sum_{\forall j \in Q_i} w_j \right), i = 1, 2, .., m \end{aligned}$$

As an example, consider the weighted graph G shown in Fig.3.9. A total of 9 cliques and 2 maximal cliques ($Q_1 = \{1,2,3\}$ and $Q_2 = \{3,4\}$) can be identified in G. Thus,

$$\begin{aligned}
\omega(G) &= \max(|Q_1|, |Q_2|) \\
&= \max(3, 2) \\
&= 3 \\
\omega_w(G) &= \max[(w_1 + w_2 + w_3), (w_3 + w_4)]
\end{aligned}$$

Observation 1. *In a given weighted graph G, the vertex weights have no influence while constructing the set of maximal cliques $\{Q_i\}$. However, given a clique set $\{Q_i\}$, the choice of the maximum weighted clique (MWC) from this set is affected by the vertex weights*

In the above example, if $W = [1,1,1,1]$, then the $MWC = Q_1$ and $\omega_w(G) = 3$. On the other hand, if if $W = [1,1,1,3]$, then $MWC = Q_2$ and $\omega_w(G) = 4$.

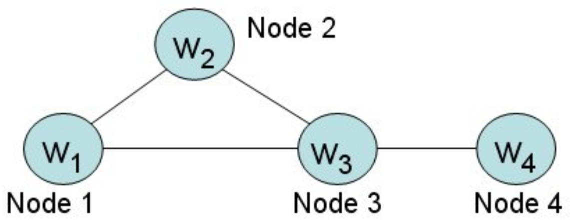


Figure 3.9. Example weighted graph to illustrate clique and maximal clique concepts. A total of 9 cliques can be identified: $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1,2\}$, $\{1,3\}$, $\{2,3\}$, $\{3,4\}$, $\{1,2,3\}$. Out of these 9 cliques, only two are maximal cliques: $Q_1 = \{1,2,3\}, Q_2 = \{3,4\}$.

For a given weighted graph G, the weighted chromatic number $\chi_w(G)$ is at least as large as the maximum weighted clique number $\omega_w(G)$.

$$\omega_w(G) \leq \chi_w(G)$$

Unfortunately, for most classes of graphs, computing the clique number is also an NP-complete problem, i.e., computing the clique number is at least as difficult as computing the chromatic number. However, while the chromatic number problem on a UDG is still NP-complete, *the clique number problem can be solved in polynomial time in UDGs* [BCD90]. Moreover, in UDGs, the clique number can be used to upper bound the chromatic number within some constant factor. Even if the UDG is weighted, it does not pose any difficulties; the polynomial time transformation preserves the polynomial time complexity of the clique number computation procedure for UDGs.

3.6 Imperfection Ratio

For weighted UDGs, the authors in [GM01] introduce the metric "imperfection ratio", $imp(G)$, defined as the supremum of the ratio of its chromatic number to its clique number. The supremum is computed over all possible weight vectors W. They also bound $imp(G)$ as:

$$imp(G) = \sup_{W} \frac{\chi_w(G)}{\omega_w(G)} \leq 2.155$$

We now seek to apply this result for practical wireless networks. Using an upper bound of $2.155\omega_w(G)$ to bound the chromatic number may be too conservative, especially if ratio of the chromatic and clique numbers for *most* cases is close to 1. This would mean overestimating the required number of time slots under optimal scheduling by more than a factor of 2, thus underestimating available bandwidth. In such cases, using 2.155 as a bound represents the "worst-case" scenario.

The authors in [GM01] mention that the $imp(G) = 1$ iff G is "Perfect", i.e., in the special case of a "Perfect Graph" [JB01], χ and ω have equal values in every induced subgraph (weighted and unweighted). They also speculated that the bound could be improved to 1.5 for non-perfect UDGs. However, even non-perfect graphs can have equal values for χ_w and ω_w. Since we are only interested in the relationship between $\chi_w(G)$ and $\omega_w(G)$ for the *given* weighted graph G and not for the induced subgraphs of G, a non-perfect graph might have equal values (or nearly so) for the clique and chromatic numbers for a vast majority of weight vectors. We investigate and quantify this possibility in the next chapter, with the goal of obtaining a practical bound on the chromatic number that is tighter than the theoretical bound.

3.7 Summary

We used the Tx-model as the model of interference in this dissertation. The Tx-model enabled the use of Unit Disk Graphs to capture the interference relationship between the various nodes in a given MHWN. For a given rate vector X in the MHWN under a specific scheduling algorithm, we computed the interference sensed by each node from the global schedule returned by the scheduling algorithm. Using these interference values, we then formulated the global capacity constraints, and showed that these constraints are both necessary and sufficient conditions for the feasibility of X.

We then specifically focused on optimal scheduling, and observed that optimal scheduling in a MHWN for a given rate vector X can be achieved by optimal weighted coloring of the associated UDG. Specifically, the span of the optimal schedule was equal to the weighted chromatic number of the UDG. Noting that the optimal coloring procedure for UDGs was NP-hard, we focused on computing bounds for the weighted chromatic number. We observed that the weighted clique number of the UDG can be computed in polynomial time, and can be used to both upper and lower bound the weighted chromatic number. Noting that the theoretical upper bound on the ratio of the weighted chromatic number to the weighted clique number may be too conservative for practical wireless network scenarios, we investigate the possibility of improving the bound for practical wireless networks in the next chapter.

Chapter 4

Bounds for Practical MHWNs

4.1 Practical Bounds on Imperfection Ratio

In this subsection, we construct distributions of the ratio of chromatic number to clique number, as opposed to a theoretical upper bound on that ratio (as in [GM01]). The general goal is to determine the likelihood that the clique number ω_w is "close to" the chromatic number χ_w for a randomly chosen weighted graph G. This study is an extension of our work in [MP07].

Mathematical analysis to provide information on the closeness of chromatic number and clique number seems very difficult to do. The other approach is to employ an exhaustive search method over all possible weight vectors for all possible UDG combinations. Clearly, the second approach is infeasible. Hence we consider slightly less expensive estimation strategies.

To begin, we recall that the $imp(G)$ is computed over all possible weight vectors W, where W represents the transmission rates of each of the nodes in a MWHN. From a practical wireless network perspective, it does not make sense to consider *all* possible weight vectors W; instead one could limit the maximum weight (transmission rate) to a reasonable value for MHWN UDGs, which may improve the bound. While an exhaustive search will still not be possible even with a bounded maximum weight, we can conduct experiments that will search through a very large number of scenarios.

To assess closeness of χ_w and ω_w, we define the measure *partial imperfection ratio (PIR)* of a weighted graph G, defined as the ratio $\chi_w(G)/\omega_w(G)$ for a given weight vector W. PIR values closer to 1 indicate very high closeness. The following is our experimental scenario: we assume our area of interest to be a disk of radius 1. We place n nodes in randomly chosen locations within the disc and any two nodes are connected by an edge if they lie within a distance of β, ($\beta = 1$) from one another. Node u is assigned an integer weight w_u that corresponds to its traffic requirements. The weights are chosen randomly having a uniform p.m.f in $1, 2, .., K$, where K corresponds to the maximum weight. To study the influence of nodal density on PIR, we varied n as 10, 25, 50, 75 and 100. Also, to study the effect of having various node traffic rates, we independently varied K as 1, 5 10, 20, 30, 40, and 50. It has to be noted that the mean weight assigned to a node in weighted UDG G is $0.5(K+1)$, and hence the mean number of nodes in transformed, unweighted UDG H is $0.5n(K+1)$. Thus, for the purposes of graph coloring using the transformation, the smallest mean size of the transformed UDG in our study is 10, and the largest is 2550.

An experiment in our case comprised 10000 trials conducted for a given (n, K) pair. For each experiment, the CDF curves corresponding to the PIR values were plotted. We used MATLAB to generate graphs in DIMACS [dim] format. For optimal coloring, we used the DSATUR program [dsa] written in C. To compute the maximum weight clique, we used the CLIQUER software [cli], also written in

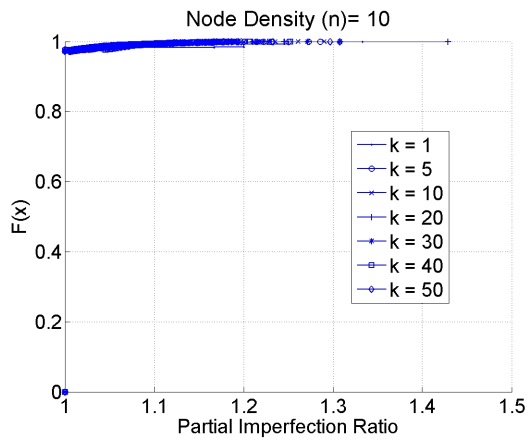
C.

Fig.4.1 shows cdf plots for the various (n, K) values. First, we focus on the variation in PIR values with respect to changes in K values. For a given value of node density n, the CDF curves for the various K values (with the exception of $K = 1$), are quite similar. The variance in the individual node weights (traffic rates) increases with increase in K. The results when compared across K seem to suggest that in a given network topology, there is a higher probability of the PIR value to deviate from the 1.0 mark as K increases, with the exception of $K = 1$, which shows a stepwise behavior. A value of $K = 1$ corresponds to a *balanced or near-balanced* load scenario, where all nodes generate the same amount of traffic.

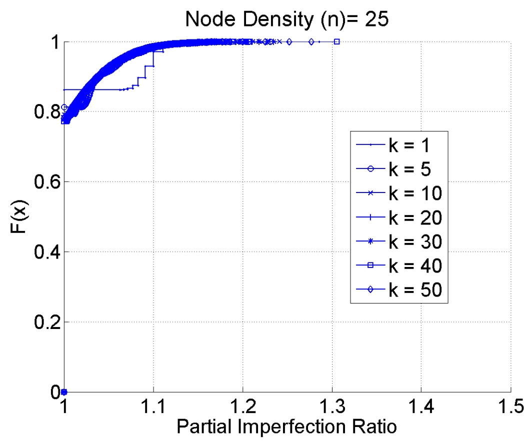
Now we focus on the variations in PIR values for changes in node density n. The number of instances when $PIR = 1.0$ (exactly) seems to steadily decrease with increase in node density. Interestingly, the PIR-spread (max PIR - min PIR) seems to *decrease* with increase in node density. That is, at higher node densities, most of the PIR values are only slightly higher than the ideally desired value 1.0. The number of outliers seem to be more pronounced at lower node densities. In fact, the maximum PIR value for each traffic rate (K) was noticed for $n = 10$. This is an encouraging result because the the bulk of PIR values for *all* scenarios are either equal to 1.0 (lower node densities) or very close to 1.0 (higher node densities).

The immediate conclusion one can draw from these plots is that the theoretical bound of 2.155 on the ratio of chromatic to clique numbers *for practical MHWN UDGs* is extremely conservative. The maximum value of PIR noticed was 1.428 for $n = 10, K = 20$.

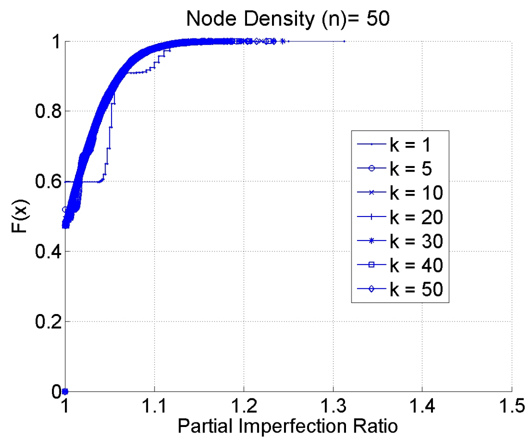
Further inspection of the plots reveals that a vast majority of the PIR values



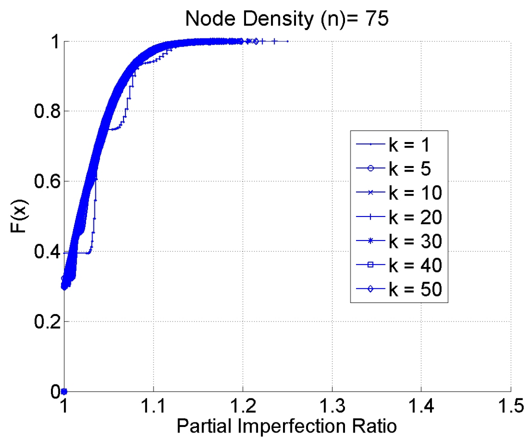
(a) Node density (n) = 10



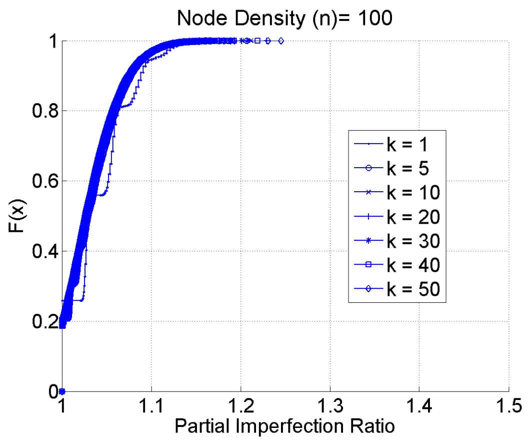
(b) Node density (n) = 25



(c) Node density (n) = 50

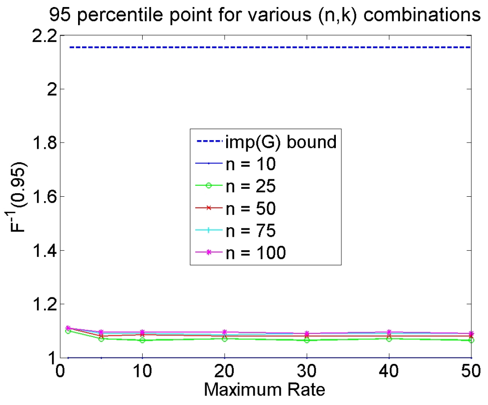


(d) Node density (n) = 75

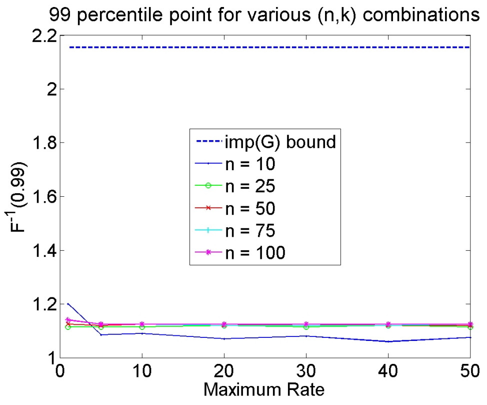


(e) Node density (n) = 100

Figure 4.1. CDF plots for Partial Imperfection Ratio (PIR) for various *(n,K)* values and $\beta = 1$



(a) 95th percentile PIR point



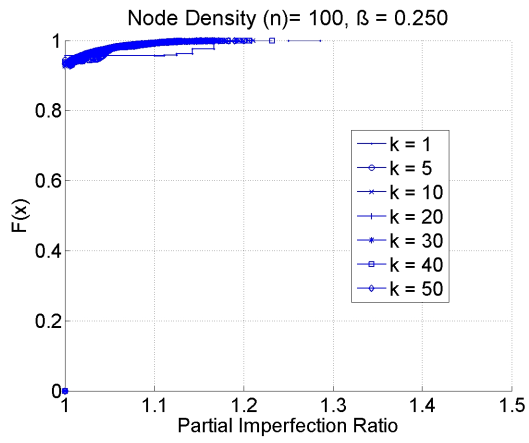
(b) 99th percentile PIR point

Figure 4.2. 95 and 99 percentile PIR values for various *(n,K)* values and $\beta = 1$

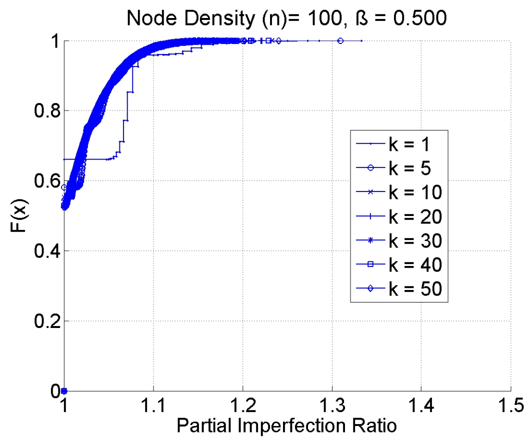
are very close to 1.0. In fact, as Fig.4.2a shows, 95% of the PIR values for all scenarios considered are less than 1.1. Fig.4.2b shows that 99% of the PIR values for all scenarios considered are less than 1.2. Also, a quick inspection of the median (50 percentile point) values across all scenarios reveals a maximum median value of 1.03. *This implies that in at least* 50% *of all the cases investigated, the PIR value deviated from the ideal value of* 1.0 *by a maximum of* 3%.

For the case with $n = 100$, we also conducted experiments by varying the value of β. This had the effect of varying the network diameter or number of "connection" hops (lower $\beta \Rightarrow$ more network hops). As β was varied as *0.25, 0.5, and 0.75*, the cdf results observed were very similar to the case with $\beta = 1$ and n = *(25, 50, 100)*, respectively. See Fig. 4.3 for cdf plots and Figures 4.4a, 4.4b for percentile plots.

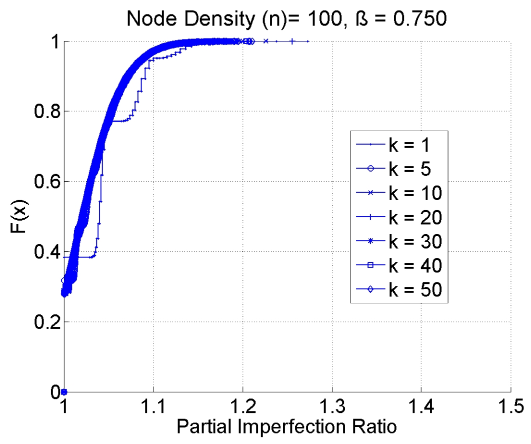
Thus, we now have evidence to believe that the theoretical bound of 2.155 very rarely holds in practice, and using this bound almost always grossly over-estimates the actual interference value and under-estimates the available wireless network capacity. Based on our investigation, we propose that *in practical wireless*



(a) $\beta = 0.25$



(b) $\beta = 0.50$

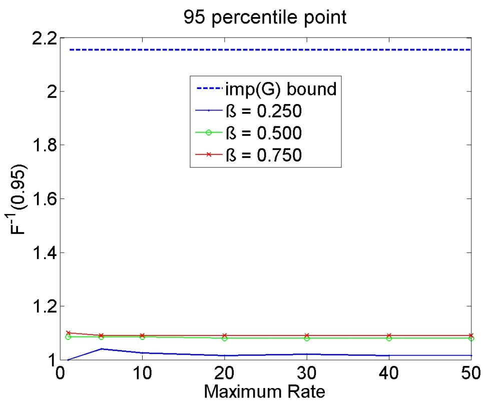


(c) $\beta = 0.75$

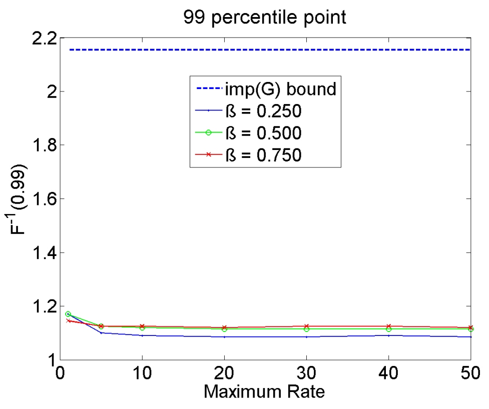
Figure 4.3. CDF plots for Partial Imperfection Ratio (PIR) for various (β, k) values for $n = 100$

UDGs, the weighted clique number ω_w *itself can be used an excellent and accurate approximation to the weighted chromatic number* χ_w :

$$\omega_w \approx \chi_w \tag{4.1}$$



(a) 95th percentile PIR point



(b) 99th percentile PIR point

Figure 4.4. 95 and 99 percentile PIR values for various (β, k) values for $n = 100$

4.2 Consequences of Practical Bound

Given a MHWN and a rate vector X, we have so far established the following for the corresponding weighted UDG G under a Tx-model of interference:

- the *global* capacity constraints are both necessary and sufficient conditions for the feasibility of X
- formulating the global capacity constraints can take exponential time due to the NP-completeness of the optimal coloring procedure for UDGs
- the span M of the optimal schedule S can be accurately approximated using the weight of G^M (the MWC of G)

While the approximation in (4.1) does help in assessing the feasibility of X, it does not help in formulating the global capacity constraints. The global capacity constraints cannot be formulated because the actual interference values for node u (e_u) are not known (our clique approximation only yields the span M, not the number of time slots in the schedule that interfere with node u). In fact, it raises the following question: if the feasibility of X can be assessed in polynomial time using the approximation in (4.1), then are the global capacity constraints even essential?

We believe that the global capacity constraints are indeed desirable based on the following argument: in an optimization problem, the goal is usually to generate a sequence of rate vectors $\{X_k\}$ until an X^* is found that optimizes the objective function. Note that the optimal point X^* resides within the feasible region defined by the complete set of problem-specific and network-imposed constraints. Since the global capacity constraints are necessary conditions for the feasibility of any X, the set of global capacity constraints formulated until some iteration k in the optimization problem defines the feasible region at least partially (we use the term "partial" because there could be more limiting constraints that we are not aware of) i.e., a future rate vector $X_{k+\tau}, \tau > 0$ that violates the any of global capacity constraints formulated until iteration $k + \tau - 1$ is *certainly* infeasible. Hence, the sequence of points generated in future can at least be guaranteed not to fall outside the partial region of feasibility defined by these global capacity constraints. On the other hand, if only feasibility was checked for each X_k, and no global capacity constraints were formulated, then there is the possibility of generating points in the future that fall outside the parital feasible region, which constitutes wasted effort. Thus, having some form of capacity constraints could potentially speed up the optimization procedure.

Since the global capacity constraints are difficult to construct, we construct a set of constraints we call the *local* or L-capacity constraints that have *similar* properties to the global capacity constraints. By similar, we mean that the L-capacity constraints have a similar form to the global capacity constraints and are both necessary and sufficient conditions for feasibility of some rate vector X. These constraints have the modifier "local" because they are constructed using the optimal transmission schedule by considering only nodes "around" the *local* neighborhood of a given node. In other words, the L-capacity constraints are constructed for subgraphs of G (one subgraph per node).

To this end, we construct one subgraph per node in G, called *Reduced Interference UDG*, which is the UDG formed by the interfering neighbors of the given node (node u is excluded from the reduced UDG), and compute the MWC for each of these graphs. The reduced UDG for node u can be represented as $G_u^R(V_u, E_u^I, W_u^R)$, where:

$$\begin{aligned} V_u &= \{v | v \in N_u^I\} \\ E_u^I &= \{uv | d(u,v) \leq I_R, \forall v \in V_u\} \\ W_u^R &= \{w_v | \forall v \in V_u\} \end{aligned}$$

As an example, Fig.4.5 shows the reduced UDG for node 4 in the interference UDG from Fig.3.6 (time slotted system). The vertices of G_4^R are $V_4 = \{2, 3, 5, 6\}$, with weights $W_4'^R$ of $[2, 2, 2, 1]$.

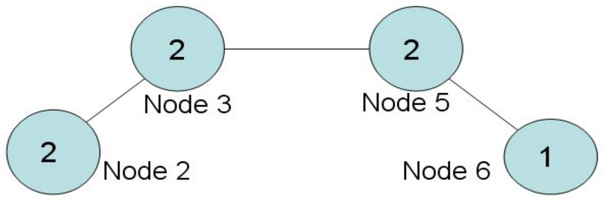


Figure 4.5. Reduced Interference UDG corresponding to node 4 (G_4^R) in 6-node chain topology.

An optimal coloring of the reduced UDG of node u produces an optimal schedule S_u of span M_u (local solution). The *local* interference for node u, denoted as e'^{l}_{u}, is simply M_u, because every slot of S_u has at least one interfering neighbor of u scheduled.

Observation 2. *In a MHWN represented by the weighted UDG $G(V, E, W)$, for a given node $u \in V$ with reduced weighted interference UDG G^R_u, the local interference rate (e^l_u) or number of interfering timeslots (e'^l_u) in the local schedule S_u of span M_u sensed under an optimal transmission scheduling scheme equals the weighted chromatic number χ_w of the reduced weighted interference graph G^R_u.*

$$e'^{l}_{u} = \chi_w(G^R_u)$$

From observation-2 the following set of *local* or L-capacity constraints can be formulated:

$$\forall u \in V : \sum_{\forall v \in N^C_u} x'_{uv} + e'^{l}_{u} = \sum_{\forall v \in N^C_u} x'_{uv} + \chi_w(G^R_u) \leq \gamma \tag{4.2}$$

The following is true for the reduced UDGs:

$$\begin{aligned} \forall u \in V : M_u &\leq M \\ 0 \leq e^l_u &\leq e_u \\ => \sum_{\forall v \in N^C_u} x_{uv} + e^l_u &\leq \sum_{\forall v \in N^C_u} x_{uv} + e_u \end{aligned} \tag{4.3}$$

As an example, consider the reduced UDG shown in Figure 4.5 corresponding to node 4. Here M_4 is 4 time slots, which is also equal to the local interference e'^{l}_{4} sensed by node 4. The global interference value for node 4 (e_4) computed from the global optimal schedule shown in Figure 3.7 also happens to equal 4 time slots (equality need not always hold).

From (4.3), it follows that the L-capacity constraints in 4.2 are *necessary* conditions for feasibility, but are not sufficient i.e., any rate vector X that satisfies the global capacity constraints (feasible X) will also satisfy the L-capacity constraints by virtue of (4.3). However, if only the local interference values $e_u^{\prime l}$ are available for a given X, then nothing can be said of the feasibility of X. In this case, the knowledge of the scheduling span M is essential to assess the feasibility of X.

We will demonstrate sufficiency of the L-capacity constraints by showning that there exists at least one L-capacity constraint (say, corresponding to node u) with high likelihood such that the following is true:

$$\sum_{\forall v \in N_u^C} x'_{uv} + e_u^{\prime l} = M \tag{4.4}$$

If such a constraint exists, then M can be computed from the following:

$$\max_{u \in V} \sum_{\forall v \in N_u^C} x'_{uv} + e_u^{\prime l} = M \tag{4.5}$$

Equation (4.5) must hold if (4.4) is true because the span M_u of the local schedule S_u corresponding to *any* L-capacity constraint is never greater than the span M of the global schedule. If the existance of an L-capacity constraint that satisifes (4.4) can be established, then M can be computed from (4.5), thus demonstrating sufficiency of the L-capacity constraints for the feasibility of a given rate vector X.

In addition, if (4.4) is true, then the following is also true by substitution of alternate notation ($M = \chi_w(G)$, $\sum\limits_{\forall v \in N_u^C} x'_{uv} = w'_u$, $e''_u = \chi_w(G_u^R)$):

$$\chi_w(G_u^R) = \chi_w(G) - w_u$$
$$e'_u = e''_u = \chi_w(G_u^R)$$

From (4.1), we know that $e''_u \approx \omega_w(G_u^R)$. Thus, we can rewrite (4.5) as follows:

$$\max_{u \in V} \sum_{\forall v \in N_u^C} x'_{uv} + \omega_w(G_u^R) \approx M \tag{4.6}$$

Observing that $M \approx \omega_w(G)$, the problem of showing the existance of an L-capacity constraint such that (4.4) is satisfied (and thus the sufficiency) with *high likelihood* reduces to one of showing the existance of at least one reduced UDG G_u^R, such that $\omega_w(G_u^R) = \omega_w(G) - w'_u$.

Lemma 2. *In an MHWN, if the interference relationship is expressed using interference UDG $G(V, E, W)$, then $\exists u \in V$ such that the reduced interference UDG corresponding to u, G_u^R, satisfies the following relationship: $\omega_w(G_u^R) = \omega_w(G) - w_u$*

Proof: Let $G^M(V^M, E^M, W^M)$ denote the (complete or fully interconnected) subgraph corresponding to the MWC of G ($\omega_w(G) = \omega_w(G^M) = \sum\limits_{\forall u \in V^M} w_u$). First we show that $\forall u \in V^M : G^M - u \subseteq G_u^R$ (the notation $A \subseteq B$ when applied to graphs implies that graph $A(V_A, E_A)$ is completely contained in graph $B(V_B, E_B)$ i.e., $V_A \subseteq V_B$ and $E_A \subseteq E_B$). Since every node in V^M is pair-wise adjacent, for a given $u \in V^M$, all nodes in $\{V^M \setminus u\}$ are adjacent to (or neigbhors of) node u . Clearly, $\{V^M \setminus u\} \subseteq V_u$. Thus, the complete subgraph corresponding to $G^M - u$ must be fully contained in G_u^R, because G_u^R is the subgraph formed by *all* neighbors of u i.e., $\forall u \in V^M : G^M - u \subseteq G_u^R$. It is equivalent to saying that:

$$\forall u \in V^M : G^M \subseteq G^R_u + u \tag{4.7}$$

Also note that $\omega_w(G^M - u) = \omega_w(G) - w_u$. We will now show that $\omega_w(G^M - u) = \omega_w(G^R_u)$ to complete the proof.

From (4.7), it is clear that the $\omega_w(G^M) \leq \omega_w(G^R_u + u)$. Similarly, noting that $G^R_u + u \subseteq G$, it can be seen that $\omega_w(G^R_u + u) \leq \omega_w(G) = \omega_w(G^M)$. Hence, $\omega_w(G^R_u + u) = \omega_w(G^M) = \omega_w(G)$. Since node $u \in V^M$ is connected to every node in $G^R_u + u$ the graph G^R_u formed by vertex removal of u will have the weight of the maximum weighted clique value reduced by w_u when compared to the weight of the maximum weighted clique value of $G^R_u + u$ i.e., $\omega_w(G^R_u) = \omega_w(G^R_u + u) - w_u = \omega_w(G) - w_u$. Hence the proof.

$\square$

It follows from lemma 2 that *all* nodes $u \in V^M$ lead to L-capacity constraints that satisfies (4.4) with high likelihood, and thus it follows that satisfying the the set of L-capacity constraints is a sufficient condition as well for the feasibility of X. As an illustrating example, consider the interference UDG G shown in Figure 3.6. The subgraph G^M corresponding to the MWC of G could be either the subgraph induced by nodes $\{2, 3, 4\}$ or nodes $\{3, 4, 5\}$, with $\omega_w(G) = 6$. Table 3.1 gives the set of nodes (interfering neighbors) that form the reduced UDG for each node. Let $G^M_u(V^M_u, E^M_u, W^M_u)$ denote the the MWC of G^R_u for some $u \in V$. Table 4.1 gives the set of nodes (V^M_u) in G^M_u for each node $u \in V$, and the corresponding ω_w values:

From Table 4.1, it can be seen that *every* node in the *all* MWCs of G $(2, 3, 4, 5)$ leads to L-capacity constraints that satisfy (4.4).

Table 4.1. Maximum Weighted Cliques for each node in the 6-node chain topology

Node u in G_u^R	w_u	V_u^M	$\omega_w(G_u^R) + w_u$
1	1	{2,3}	5
2	2	{3,4}	6
3	2	{2,4}, {4,5}	6
4	2	{2,3}, {3,5}	6
5	2	{3,4}	6
6	1	{4,5}	5

Due to observation-2, (4.1), and the practical bounds results, the following holds:

$$\begin{aligned} \forall u \in V : e_u'^l &\approx \sum_{\forall v \in V_u^M} w_v' && (4.8) \\ &= \sum_{\forall v \in V_u^M} \sum_{k \in N_v^C} x_{vk}' \\ \Rightarrow \sum_{\forall v \in N_u^C} x_{uv}' + e_u'^l &\approx \sum_{\forall v \in N_u^C} x_{uv}' + \sum_{\forall v \in V_u^M} \sum_{k \in N_v^C} x_{vk}' && (4.9) \end{aligned}$$

In the context of optimization problems, we can use the L-capacity constraints in place of the global capacity constraints to enforce feasibility of a given rate vector X. Due to (4.1), we can construct the L-capacity constraints in polynomial time. Also, each of these L-capacity constraint is a *linear* inequality, since $e_u'^l$ (and thus e_u^l) can be expressed in linear form (as shown in (4.9)). A polynomial-time linear inequality (capacity constraint) formulation procedure is very attractive in the context of solving optimization problems.

$LCONSTR$ is the polynomial-time algorithm that takes the graph representation G of the MHWN and the rate vector X as input and returns the set of L-capacity constraints at X for G (one constraint per node). Each L-capacity constraint is nothing but the constraint formed due to the MWC of the corresponding node's reduced UDG and that node's transmission rates. The MWC for each node is computed by invoking the polynomial time algorithm from [BCD90] (let us call it $MAXWTCLIQUE$ for convenience). Note that the vertex weights need to be integers; if the weights in the original problem are real numbers, then the weights need to be appropriately scaled to produce integer weights, such that the integer weights are in the same proportion to one another as the original weights. After the weighte clique number computation procedure is executed, an inverse scaling procedure needs to be applied on the resulting weighted clique numbers to recover the weighted clique numbers corresponding to the original problem with non-integer weights.

Algorithm 1 LCONSTR (G , X)

```
1: for all $u$ such that $u \in V$ do
2:     Form $G_u^R(V_u, E_u^I, W_u^R)$ for node $u$
Require: $w_v^R \in W_u^R \quad \forall v \in V_u$ are integers
3:     $mwc[u] \Leftarrow MAXWTCLIQUE(G_u^R)$
Ensure: Restore $w_v^R \in W_u^R \quad \forall v \in V_u$ to original (non-integer) state if necessary
4:     $e[u] \Leftarrow \sum_{\forall v \in mwc[u]} w_v^R$
5: end for
6: return $mwc$, $e$
```

There is another potential advantage of using the L-capacity constraints, instead of simply finding the value of M using the MWC of G. The clique finding algorithm in [BCD90] has a time complexity of $O(m^{4.5})$, where m is the number of vertices in the UDG on which the maximal clique is found. If we find construct the clique constraint using the entire UDG G with n nodes, then the complexity of the procedure is $O(n^{4.5})$. On the other hand, finding the MWC of G using the reduced UDGs has a run-time complexity of $O(n\Delta^{4.5})$, where Δ is the maximum degree (maximum number of adjacent vertices for any node) of G. This method is superior to the original method if $n > \Delta^{\frac{4.5}{3.5}}$, which holds good in networks that are not very dense or near complete graphs.

4.3 Summary

In this chapter, through an empirical study, we demonstrated tighter bounds on the ratio of the weighted chromatic number to the weighted clique number in UDGs representing practical MHWNs. In particular, we concluded that the weighted clique number can be used as an excellent approximation to the weighted chromatic number for practical MHWNs. Observing that this approximation does not aid in formulating global capacity constraints, we introduced the notion of local or L-capacity constraints, which can be formulated in polynomial time due to the clique number - chromatic number approximation introduced in this chapter. We proved that the L-capacity constraints are both necessary and sufficient conditions for the feasibility of a given rate vector X, and as a result can be used in lieu of the global capacity constraints. In addition, the L-capacity constraints are linear in nature, which makes them attractive for optimization problems. In the next chapter, we will develop an efficient optimization framework that will make use of the L-capacity constraints to solve flow-related optimization problems in MHWNs.

Chapter 5

Optimization Framework for MHWNs

5.1 Network Flow Problems in MHWNs

In section 2.5, we saw that network flow problems arising in MHWNs are difficult to solve by direct application or extension of techniques from the wired domain. We revisit the $MAXFLOW$ problem, which deals with finding the maximum amount of *flow* between a set of source-destination pairs in a given network. We chose this problem becuause it is a very popular flow problem in wired networks and has received extensive treatment in literature. The most obvious way of solving this problem is by casting the problem as a Linear Programming (LP) problem and solving it using standard LP techniques [Lue84]. However, LP techniques fail to take advantage of the special structure of the flow problem, and are in most cases not very efficient. Several specialized algorithms such as Ford-Fulkerson [LD56], Edward-Karp [EK72], etc. exist, which are more efficient than the LP solution.

Fig. 2.9 shows an LP formulation for the $MAXFLOW$ problem in a MHWN, which has been modeled as a UDG with vertex set V, and set of communicating and interfering neighbors for a given node i are given by N_i^C and N_i^I, respectively. We assume a single flow of value f exists between source node r and sink node s. If a *transmission* link exists between two nodes u and v, then the flow on the link uv is given as x_{uv}. These link flows are represented in the form of a column vector X of length equal to the number of links l. In this example, we assume that an arbitrary MHWN of n nodes is given with a channel capacity of γ.

Maximize f

Subject to

1. *Flow balance equations*

$$\begin{aligned}
\sum_{\forall j \in N_r^C} x_{rj} - \sum_{\forall j \in N_r^C} x_{jr} - f &= 0 \quad \textit{(source)} \\
\sum_{\forall j \in N_i^C} x_{ij} - \sum_{\forall j \in N_i^C} x_{ji} &= 0 \quad \forall i \in V \backslash \{r, s\} \\
\sum_{\forall j \in N_s^C} x_{sj} - \sum_{\forall j \in N_s^C} x_{js} + f &= 0 \quad \textit{(sink)}
\end{aligned}$$

2. *Capacity constraints*

$$\sum_{\forall j \in N_i^C} x_{ij} + e_i \leq \gamma, \quad \forall i \in V$$

$$0 \leq x_{ij} \leq \gamma, \quad \forall i \in V, j \in N_i^C$$

Figure 5.1. LP formulation for MAXFLOW in MHWNs

The $MAXFLOW$ formulation shown in Fig.5.1 is quite similar to the one for wired networks shown in Fig.2.9, except for the capacity constraints. The capacity constraints for $MAXFLOW$ in MHWNs include the interference term e_i, which is absent in wired networks. We already saw that the capacity constraints are very expensive to formulate in MHWNs; instead, we use the L-capacity constraints that can be formed in polynomial time (due to (4.2) and (4.8)). We already showed that for UDGs, the L-capacity constraints are both sufficient and necessary for the feasibility of X. The e_i^l values can be computed by invoking the $LCONSTR$ function call whenever a new point (link flow vector) X is computed. Recall that due to (4.8), e_i^l can be expressed as a linear combination of a subset of node weights w. More specifically, if there are m maximal cliques (Q's) that can be identified around node i, then from (4.8):

$$e_i^l \approx \max_k \left(\sum_{\forall j \in Q_k} w_j \right), k = 1, 2, .., m$$

The number of variables in the $MAXFLOW$ formulation is the sum of number of links in the communication UDG (communicating or transmission links) and the number of source-sink flows. The interference UDG is only used to determine which nodes (and thus communicating links) are in interference for the purpose of computing the interference values $\{e_i^l\}$; the interference UDG does not influence the number of variables in the $MAXFLOW$ formulation.

Since the MWC can change based on the vector X (LP solution point), e_i^l does not have a pre-determined expression (form); it could be any one (or more) of the m cliques based on X (see observation.1 and example following). Unfortunately, standard LP algorithms such as Simplex [Dan51] require constraints to be *explicitly* listed, i.e., to have a fixed form. Hence these algorithms do not permit invoking the $LCONSTR$ function call at each iteration to determine the form and value of e_i^l. To list all the capacity constraints explicitly, one needs to list or enumerate all the m maximal cliques. The *clique enumeration* [MM65] problem is a well-known *NP-Hard* problem in graph theory for most graphs (including

UDGs), because the number of cliques could be exponentially many. Thus, using standard LP techniques to solve the $MAXFLOW$ problem in MHWNs is infeasible because it requires solving an *NP-hard* problem (clique enumeration) (the $LCONSTR$ algorithm does not employ clique enumeration to compute the MWC; instead it exploits the geometry of the UDGs to compute the MWC in polynomial time). Usually, this situation is tackled by listing a large subset of the cliques as in [JPPQ03] or listing all sets that can be approximated as cliques (super-maximal cliques) [GWG05] to formulate the constraints, and then solving using standard LP techniques.

These approximation approaches to clique listing may not always perform well. For example, the total solution time for the $MAXFLOW$ problem in [JPPQ03] is large even for moderate sized networks. For a given network, the total runtime increased as a non-linear function of the number of constraints and a large number of constraints need to be listed for good accuracy. On the other hand, though the approach in [GWG05] lists all super-maximal cliques in polynomial time, it lists a large number of cliques, which leads to increased optimization run-time and memory requirements. The accuracy of the optimal solution might be affected due to the use of super-maximal cliques as approximations to the actual cliques. The super-maximal cliques are supersets of actual cliques, and might lead to over-estimation of interference. Thus, both these methods have limited scalability.

We take a novel approach in order to leverage the polynomial-time $LCONSTR$ algorithm. Instead of listing the constraints a priori, we propose a *discover-as-you-go* approach to discover constraints as needed, and integrate this approach with the well-known Rosen's gradient projection algorithm [Ros60] to solve the $MAXFLOW$ problem. We again emphasize that our solution method is not restricted only to the $MAXFLOW$ problem. We chose the $MAXFLOW$ problem because it is well-known, easy to understand, and serves to illustrate the difficulty posed by the interference (e_i^l) quantity that appear in the capacity constraints of MHWN flow problems. These constraints represent a physical limitation in MHWNs (network-imposed constraints) and will appear in most MHWN flow problems. Our solution method can be applied to all such problems.

5.2 Discover-as-you-go Approach ($DAYG$)

Our discover-as-you-go ($DAYG$) approach is inspired by the *Active set theorem* [Lue84], which allows us to solve an inequality constrained optimization problem by constructing an equality constrained problem at each iteration. The equality constrained problem is constructed from the original problem by considering only a subset of the *active* inequality constraints. An active inequality is one that is met with equality *for a given point X*. The inactive constraints at iteration k (point X_k) do not limit the search direction and hence may be ignored at that iteration. The optimal solution found using the active set approach is an optimal solution to the original problem as well. Interested readers are referred to [Lue84] for an in-depth treatment on the active set approach.

The constraints which remain inactive during the entire optimization procedure play no role under the active set approach, and any effort expended in discovering those constraints can be considered to be wasted effort. Moreover, we argue that the number of active constraints during the course of the optimization procedure is likely to be much smaller than the total number of possible constraints (cliques), given that the total number could potentially scale in an exponential fashion with the number of nodes. This intuition, if true, could lead to dramatic savings in memory requirements. Moreover, we would likely realize significant run-time savings if only active or near-active constraints (cliques) are discovered at every new point X_k, provided we have an inexpensive algorithm for constraint discovery. This is where the $LCONSTR$ algorithm comes into play.

The core of the $DAYG$ approach is the polynomial-time $LCONSTR$ algorithm. For a given graph G and a point X_k, $LCONSTR$ "discovers" the set of L-capacity constraints for X_k in polynomial time. For convenience, we restate the L-capacity constraints:

$$\forall u \in V : \sum_{\forall v \in N_u^C} x'_{uv} + e_u'^l \leq \gamma \tag{5.1}$$

Using (4.9), the summation can be represnted in vector notation as $a_u X_k$, where a_u is an indicator (row) vector of length equal to the number of links (l). The entries in a_u corresponding to the links in summation take a value of 1, while the remaining entries are assigned a value of 0. These vectors can be arranged in the form of a matrix A, which corresponds to the set of linear inequalities for the optimization problem. The matrix A can be augmented with constraints discovered at each iteration. Notice that the constraints "discovered" at point X may not be unique, i.e., more than one node could have the same MWC or some constraints discovered at point X_k may already have been discovered at a previous point. It is sufficient to augment the A matrix with only new, unique constraints.

A newly discovered constraint (a_u) for a given node u could be in one of three states for a given X_k: active ($a_u X_k - \gamma_u = 0$), inactive ($a_u X_k - \gamma_u < 0$), or violated ($a_u X_k - \gamma_u > 0$). Once a constraint has been constructed, it is trivial to check if the constraint is in one of these three states. We now integrate the discover-as-you-go approach with the well-known active set approach, the *Gradient Projection*

Method (GPM) developed by Rosen [Ros60]. We call this the *Modified* gradient projection method ($MGPM$).

5.3 Modified Gradient Projection Method

$DAYG$ could conceivably be integrated with any primal method of optimization such as the Simplex method, GPM, etc. The GPM was favored over the Simplex method because, as we will see later, the memory requirements of GPM with respect to the problem size scale much better than the requirements of Simplex method. Moreover, the dynamic constraint discovery of $DAYG$ implies that the constraint matrix will need to be resized for every new constraint discovered. For GPM, the matrix needs to be resized only along the rows; for Simplex, resizing needs to occur both along rows as well as columns because the Simplex method introduces one new variable for every inequality constraint.

We will first briefly describe the GPM (material adapted from [Lue84]) that can be used to solve optimization problems with linear constraints, followed by the description of MGPM. GPM is based on a pure active set strategy, and is motivated by the ordinary method of steepest descent for unconstrained problems. In GPM, the negative gradient of the objective function is projected onto the *working surface* in order to define the direction of movement. Consider problems of the following form that have m linear inequalities and n linear equalities:

$$\begin{aligned} \text{minimize} \quad & f(X) \\ \text{subject to} \quad & a_i X \leq b_i, \quad i = 1, 2, ..., m \\ & a_i^e X = b_i^e, \quad i = 1, 2, .., n \end{aligned}$$

At a feasible point X_k, there will be a certain number $q \leq m$ of the active inequality constraints satisfying $a_i X_k = b_i$ and $m-q$ inactive constraints satisfying $a_i X_k < b_i$. All equality constraints are active at every iteration. The working set $W(X_k)$ is the set of active constraints at X_k. The *GPM* seeks a feasible direction vector $\mathbf{d}$ satisfying $\nabla f(X)\mathbf{d} < 0$, so that movement along $\mathbf{d}$ will decrease the objective function f. Let A_q be defined as the subset of $W(X_k)$, such that the A_q is composed of linearly independent rows. The *GPM* considers feasible directions that satisfy $A_q\mathbf{d} = 0$, so that all the active constraints remain active. This requirement amounts to requiring that $\mathbf{d}$ lie in the tangent subspace M defined by the working set of constraints. *GPM* uses the projection P of the negative gradient $-\nabla f(X_k)$ onto this subspace M as the desired direction $\mathbf{d\_k}$.

Given a non-zero feasible direction, a new point X_{k+1} is computed by moving a "distance" (step size) of $\alpha \geq 0$ along $\mathbf{d}$ such that $X_{k+1} = X_k + \alpha\mathbf{d}$ is feasible, and yet realizes the maximum reduction in f along $\mathbf{d}$. This process is repeated until $\mathbf{d} = 0$. If the feasible direction is zero, then it means that no further reduction can be achieved with the current working set and the current point X_k is *potentially* an optimal point. This point is checked for the Karush-Kuhn-Tucker (KKT) conditions [Kar39, KT51] by computing the vector of Lagrange multipliers λ for working constraints at X_k. If $\lambda \geq 0$, then X_k is the optimal point. Otherwise, the active inequality constraint corresponding to the most negative λ is dropped from the working set $W(X)$ and the iteration is repeated.

The *GPM* is a *feasible set* algorithm, which requires that the points X_k generated at each iteration be feasible. The computation of the feasible direction $\mathbf{d}$ at a given X_k requires only the current working set (active constraints); on the other hand, the distance α moved along $\mathbf{d}$ requires knowledge of the inactive constraints. When the GPM is applied to problems where all the constraints are not known *a priori*, then the α computed using the current set of inactive constraints may yield an infeasible X_{k+1} due to undetected constraints inactive at X_k.

We overcome this problem by modifying the GPM to yield $MGPM$. In $MGPM$, a potential new point $\hat{X}_{k+1}$ is computed as $\hat{X}_{k+1} = X_k + \alpha\mathbf{d}$, and is checked for feasibility by invoking a *separation oracle* [JPPQ03] at $\hat{X}_{k+1}$. The separation oracle is a procedure that tells if a given point violates any constraints, and returns the violated constraints, if any. Since $\hat{X}_{k+1}$ is feasible with respect to the current working set A_q (already discovered constraints), any infeasibility of $\hat{X}_{k+1}$ will be due to a subset of undetected constraints at $\hat{X}_{k+1}$.

For MHWN flow problems with an underlying UDG, the $LCONSTR$ algorithm is used to determine new (undetected) constraints. The newly discovered constraints are added to the current inequality constraint set (A matrix), and the states of the constraints are checked (separation oracle). If any newly discovered constraint is in the violated state, then the point $\hat{X}_{k+1}$ is infeasible. The infeasible point is discarded and a new $\hat{X}_{k+1}$ that is feasible w.r.t the current set of constraints is found. This process of computing and discarding intermediate points $\hat{X}_{k+1}$ is repeated until a feasible $\hat{X}_{k+1}$ results.

Note that for every infeasible $\hat{X}_{k+1}$, the inequality constraint set is guaranteed to be augmented with the violated (and newly discovered) constraints. This has the effect of yielding different step sizes (α) for each successive infeasible $\hat{X}_{k+1}$, eventually leading to a feasible $\hat{X}_{k+1}$. Once a feasible $\hat{X}_{k+1}$ is found, then the optimization procedure moves to the new point $X_{k+1} = \hat{X}_{k+1}$. It can be observed that the $LCONSTR$ algorithm only returns a single dominant MWC for a given node at any given point X_k. Now consider the case where there is more than one

MWC for a given node u at a given X_k, i.e., more than one clique with total weight equal to ω_w of the reduced interference UDG of u. Though $LCONSTR$ returns only one MWC per node at X_k, the procedure of repeated computing and discarding intermediate points $\hat{X}_{k+1}$ ensures the feasibility of the generated X_{k+1}.

The $MPGM$ algorithm is shown Algorithm 2. All steps in $MPGM$, except steps 7 - 13, are identical to the GPM algorithm (adapted from [Lue84]). The additional steps in $MPGM$ discover new constraints using $LCONSTR$ and perform the separation oracle functionality to ensure feasibility of the output points X_k.

5.4 Evaluation

In this section, we evaluate the performance of the discover-as-you-go ($DAYG$) scheme as embodied in $MGPM$ by solving a single-flow $MAXFLOW$ problem. The DAYG approach pertains to discovering constraints dynamically during the course of optimization, as opposed to discovering the constraints a priori. To study the gains of $DAYG$, we compare it with the following two approaches that formulate constraints *a priori*:

- $RAND$: Method of constraint listing in [JPPQ03], in which $MAXEFFORT$ iterations of constraint listing are performed, where $MAXEFFORT$ depends on required degree of accuracy. A value of $MAXEFFORT = 150000$ was used in [JPPQ03] to solve the $MAXFLOW$ with high accuracy in a grid topology of size up to 11 (121 nodes). We use this method to list cliques for the topology under consideration, with a minor variation - instead of specifying a fixed $MAXEFFORT$ for all topologies, we specify $MAXEFFORT = min(max(l * \eta, 10000), 150000)$, where l is the number

Algorithm 2 MPGM (G, f, A_e, b_e, A, b, X_{init})

```
1:  $k \Leftarrow 0; X_k \Leftarrow X_{init}$
2:  Find subspace of active constraints $M$, and form $A_q$, $W(X_k)$
3:  $\mathbf{P} = I - A_q(A_q^T A_q)^{-1} A_q^T; \quad \mathbf{d} = -\mathbf{P}\nabla f(X_k)$
4:  if $\mathbf{d} \neq 0$ then
5:      Find $\alpha_1$ and $\alpha$ achieving, respectively
        $max\{\alpha_1 : X_k + \alpha_1\mathbf{d}$ is feasible $\}$, and
        $min\{f(X_k + \alpha\mathbf{d})$: $0 \le \alpha \le \alpha_1$ $\}$
6:      $\hat{X}_{k+1} = X_k + \alpha\mathbf{d}$
7:      $[MWC, e] = LCONSTR(G, \hat{X}_{k+1})$
8:      Form matrix of newly identified constraints $A_N$ from $MWC$
9:      Augment inequality matrix $A \leftarrow A \cup A_N$ and corresponding $b$ vector
10:     if $A\hat{X}_{k+1} \le b$ then
11:         $X_{k+1} \leftarrow \hat{X}_{k+1}$
12:         $k \leftarrow k + 1$
13:     end if
14:     GOTO step 2
15: else
16:     find $\lambda = -A_q(A_q^T A_q)^{-1} A_q^T \nabla f(X_k)$
17:     if $\lambda_j \ge 0 \ \forall j \in W(X_k)$ then
18:         $X_k$ satisfies KKT conditions. GOTO step 24
19:     else
20:         delete row from $A_q$ corresponding to the inequality with most negative
            component of $\lambda$ (and drop corresponding constraint from $W(X_k)$
21:         GOTO step 3
22:     end if
23: end if
24: return $X_k$
```

of links, and η is the number of iterations per link. This allows the number of iterations to scale with the problem size. Another important difference from [JPPQ03] is that we make *RAND* UDG-aware i.e. we use *RAND* to list cliques in a node-level conflict graph, instead of a link-level conflict graph as used in [JPPQ03].

- *CLIQUDG*: Method of clique listing for UDGs described in [GWG05], where a bounded maximum number ($O(m\Delta)$, m = number of links, Δ

= maximum number of neighbors or node degree) of maximal cliques and super-maximal cliques as approximations to maximal cliques are listed.

Under $DAYG$, the $MGPM$ is employed to solve the optimization problem. Under $RAND$ and $CLIQUDG$, the constraints are first discovered for the given topology using $RAND$ and $CLIQUDG$ respectively, and the appropriate LP is generated. The LP is then solved using the GPM (active-set approach) to ensure fair comparison with the $DAYG$ ($MGPM$) approach which uses GPM as the underlying optimization algorithm. Integrating $DAYG$ to into other optimization approaches is scope for future work.

In addition to these three methods, we also use a fourth method called $IDEAL$, which serves as the baseline case for performance evaluation of these algorithms. The $IDEAL$ method takes *all* of the constraints discovered in $DAYG$ and uses GPM to solve the $MAXFLOW$ problem. The $IDEAL$ method is "ideal" because it does not incur any constraint discovery overhead (*a priori* or dynamic), and the number of constraints is the minimum of the other three methods. Thus, the performance of the other three algorithms in terms of run-time or memory requirement, when using an active-set strategy (GPM), cannot exceed that of the $IDEAL$ method.

All experiments were conducted using MATLAB 7.5 (R2007b). To compute the MWC of the UDG in $LCONSTR$, the C implementation provided by CLIQUER [cli] was cross-compliled to be used with MATLAB. However, CLIQUER does not implement $MAXWTCLIQUE$, which is the MWC algorithm meant for UDGs (from [BCD90]); instead it is an efficient general-purpose MWC computational tool. We use CLIQUER because the implementation of $MAXWTCLQUE$ was not available, and the implementation required considerable programming effort. In any case, for UDGs, it is expected that $MAXWTCLIQUE$ will perform better than the algorithm in the CLIQUER software. The $MGPM$ was implemented by modifying the *linprog* and *qpsub* functions in MATLAB's optimization toolbox. For $CLIQUDG$, the MATLAB implementation provided by the authors in [GWG05] was used. For $RAND$, we wrote our own implementation in MATLAB based on the algorithm described in [JPPQ03]. The experiments were run on a machine running CentOS R5 Linux, with a Intel(R) Xeon(TM) CPU 2.66GHz processor and 4 GB of RAM.

For performance metrics, ideally, we would like to compare the total time-complexity and memory requirements of the entire optimization for each of these methods. Unfortunately, as is the case with most optimization procedures, the time-complexity of GPM is not known, i.e., the total run-time cannot be bounded as a function of input size. Thus, we resort to comparing actual measurements of run-time instead of time-complexities. It has to be emphasized that when dealing with run-time measurements, the absolute numbers do not allow meaningful interpretation. However, the relative orders of magnitude of the measurements is a good indicator of relative algorithm efficiency for comparison purposes.

The performance metrics under consideration are *constraint discovery time (CDT), total run-time (TRT),* and *number of constraints (NC)*, compared against input size (number of nodes). Since each constraint is represented in vector form (a), the amount of memory required to represent any given constraint is a constant for a given topology under all schemes, and the total amount of memory required to represent the constraints can be directly derived from NC.

We solved the $MAXFLOW$ problem in two types of topologies - grid topology and random topology. In the grid topology of size k, k^2 nodes are arranged in the form of a k x k grid. The source and sink were placed at diagonally opposite sides of the grid. The random topology was constructed by distributing nodes randomly in a $2000m$ x $2000m$ area. To ensure a connected (non-partitioned) network, the following procedure was adopted:

1. Place the source node at the location $(0, 0)$, which corresponded to one of the corners of the simulation area.

2. For every node u to be placed, pick a point (X, Y) at random, where $X, Y \sim U[0, 2000]$. If $d_{uv} \leq T_R$ for some v that has already been placed, then it implies that the (X, Y) ensures that node u is connected to the network, and is a valid location for node u (connectivity criterion). If any (X, Y) does not satisfy connectivity criterion, discard it and pick new (X, Y). Repeat procedure of discarding and picking new points as many times as necessary until connectivity criterion is satisfied for node u.

3. Before placing the sink (last) node, the node v that is farthest from the source node is identified. The sink node is placed randomly within T_R of node v. This step introduces a bias towards longer hop lengths as opposed to shorter hop lengths. Longer hop lengths are desirable to study the effects of a multi-hop path.

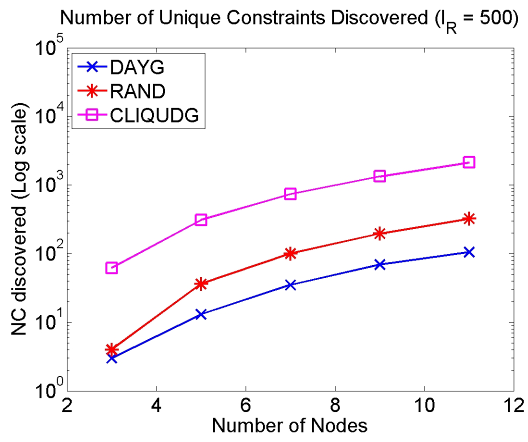
The communication range (T_R) of the nodes was taken to be $250m$ for all experiements. The experiments were conducted for two interference patterns: $C_R = 250m$ and $C_R = 550m$, leading to $I_R = 500$ and $I_R = 800m$. For $I_R = 800m$, the interference area is more than 2.5 times the area when $I_R = 500m$, thus potentially including more interfering nodes (and links). That is, the maximum (and average) node neighborhood degree (Δ) in the interference UDG is higher with high probability.

5.4.1 Grid Topology

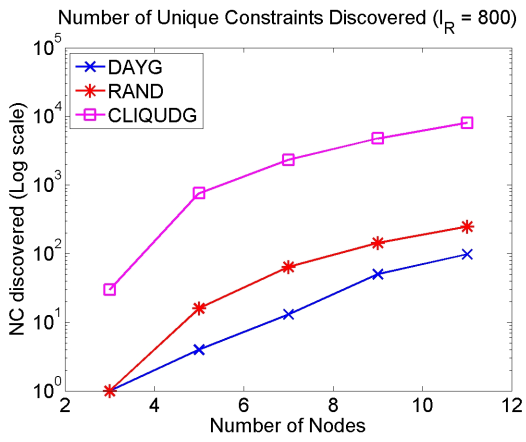
The $MAXFLOW$ problem was solved in grids of sizes 3, 5, 7, 9 and 11. Adjacent nodes in the grid are within communication range of one another (adjacent node distance = 200m), while diagonally opposite nodes are not. All methods converged to the same optimal solution for a given grid size and interference pattern. For $RAND$, the $MAXEFFORT$ heuristic used seemed sufficient to capture all the required constraints. For $CLIQUDG$, no super-maximal cliques were discovered, so the cliques discovered were actual maximal cliques.

For $I_R = 500m$, Fig.5.2a compares the total number of constraints (NC) discovered by each method, Fig.5.2c compares the CDT and Fig.5.2e compares the TRT for each method. Note the log scale on the vertical axis of each plot.

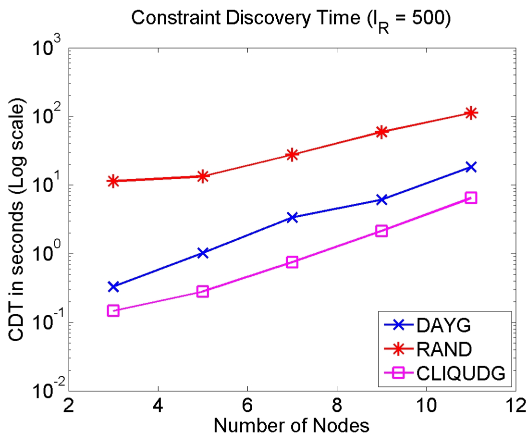
$DAYG$ consistently discovers the least number of constraints, and hence requires the least amount of memory, while $RAND$ discovers roughly twice the number of constraints than $DAYG$. While the number of constraints discovered in $CLIQUDG$ is more than one order of magnitude larger when compared to $DAYG$, it performs the best in terms of CDT. $CLIQUDG$ has a lower CDT than $DAYG$ because $CLIQUDG$ is executed only once to discover all the constraints, while $DAYG$ has to execute $LCONSTR$ every time the optimization migrates to a newer point. As noted in [Lue84], in rare cases, the GPM optimization procedure might “zigzag”, leading to an excessive number of executions of the $LCONSTR$ procedure, and thus unacceptable CDT. $RAND$ shows the highest CDT because of the large number of iterations required for clique discovery.



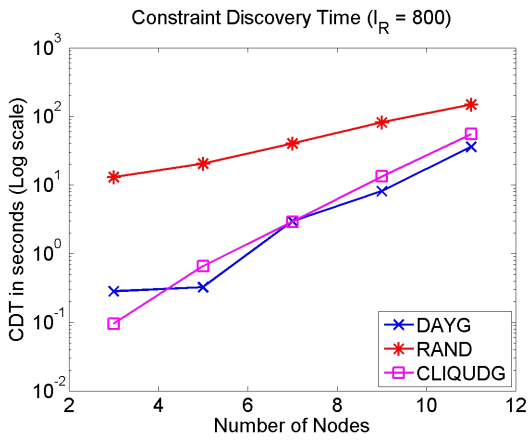
(a) NC for $I_R = 500m$



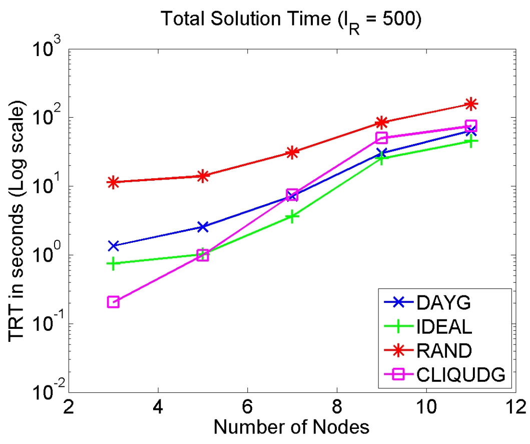
(b) NC for $I_R = 800m$



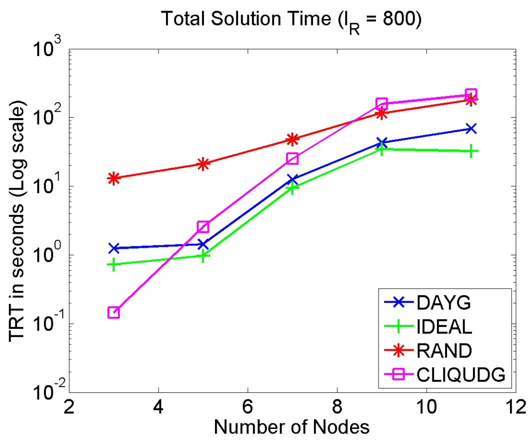
(c) CDT for $I_R = 500m$



(d) CDT for $I_R = 800m$



(e) TRT for $I_R = 500m$



(f) TRT for $I_R = 800m$

Figure 5.2. Performance Plots for Grid Topology

However, it is apparent from the NC results that not every iteration of $RAND$ produces a new constraint. $CLIQUDG$ outperforms $RAND$ in CDT because the algorithm in $CLIQUDG$ is highly tuned and specific to UDGs while $RAND$ is more general-purpose. For TRT performance, as expected, the $IDEAL$ method has the lowest value for all scenarios, except for grid size of 3, where $CLIQUDG$ is the lowest. For grid size of 3, $CLIQUDG$ seems to benefit from having a larger number of constraints than $IDEAL$ and quickly discovers the optimal solution. $CLIQUDG$ and $DAYG$ show similar TRT values, which is surprising because the CDT in $CLIQUDG$ is significantly lower when compared to $DAYG$. However, the CDT is only a small fraction of TRT, which significantly de-emphasizes the CDT gains of $CLIQUDG$ over $DAYG$. The TRT in $RAND$ exceeds the other methods by at least 0.5 orders of magnitude for all cases.

For $I_R = 800m$, Fig.5.2b compares the total number of constraints (NC) discovered by each method, Fig.5.2d compares the CDT and Fig.5.2f compares the TRT for each method. The NC for $DAYG$ and $RAND$ shows little variation from $I_R = 500m$, while the NC for $CLIQUDG$ shows a significant increase when compared to $I_R = 500m$, and differs from the other two methods by more than 2 orders of magnitude. This represents significant memory requirement for $CLIQUDG$ when compared to the other methods. The NC in $CLIQUDG$ is higher in $I_R = 800m$ due to a higher value maximum node degree Δ (number of cliques = $O(m\Delta^2)$). The CDT and TRT trends are similar to what was observed for $I_R = 500m$. Overall, the CDT in $I_R = 800m$ for all three methods is higher when compared to their respective values in $I_R = 500m$. The TRT values for all three methods for $I_R = 800m$ show a slight increase when compared to their respective values in $I_R = 500m$. However, the separation in TRT between ($DAYG$,

$IDEAL$) and ($RAND$, $CLIQUDG$) is higher in $I_R = 800m$ when compared to $I_R = 500m$

Overall, the grid topology results are very favorable for $DAYG$. The $DAYG$ comfortably outperforms $RAND$ in all aspects. $DAYG$ also shows a slightly better TRT when compared to $CLIQUDG$, yet consuming much less memory (smaller value of NC). $DAYG$ also shows very good scalability with respect to memory as can be seen from the NC plots for the two interference patterns. The lower NC value in $DAYG$ does not translate to lower CDT, as can be seen from the CDT plots. The higher CDT value is due to repeated invocation of $LCONSTR$, and each invocation of $LCONSTR$ may not result in new constraints being discovered. Since NC only counts unique constraints discovered, *the low value of NC in $DAYG$ seems to corroborate the intuition behind $DAYG$ that the most dominant cliques hop around a small subset of cliques.* This also implies that significant savings in CDT can be realized, if one can reduce the number of "wasteful" invocations of $LCONSTR$, i.e., invocations of $LCONSTR$ that do not lead to discovery of new constraints. This is potential for future work.

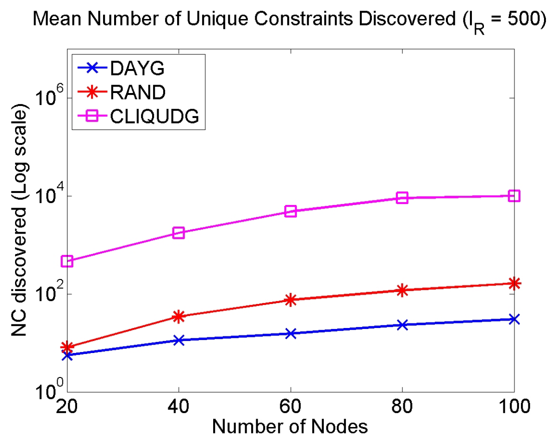
5.4.2 Random Topology

The total number of nodes in the random topology was varied as 20, 40, 60, 80 and 100, and each data point was computed as a mean over 10 simulation runs. Overall, for both $I_R = 500m$ and $I_R = 800m$, the results followed similar trends as observed for the grid topology. $DAYG$ showed the smallest value for NC when compared to $RAND$ and $CLIQUDG$, leading to substantial reduction in memory usage for both I_R values. The NC of $CLIQUDG$ was at least 2 orders of magnitude higher than $RAND$ and $DAYG$, thus requiring very high amounts of memory, which limited the scalability of that method.

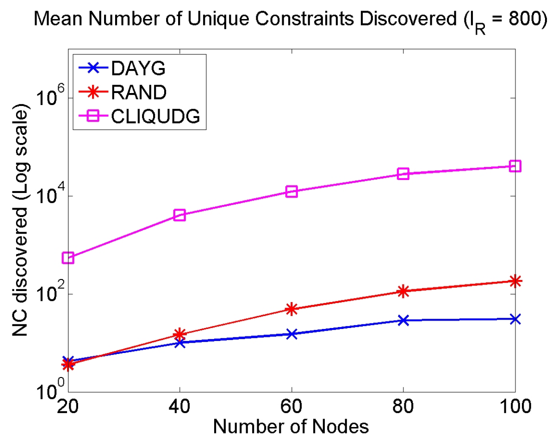
For $I_R = 500m$, $CLIQUDG$ was the fastest in terms of CDT, but the overall solution time was quite similar to $DAYG$ and $IDEAL$. For $CLIQUDG$ and $DAYG$, the CDT is a small fraction of the TRT, and thus the gains of $CLIQUDG$ over $DAYG$ w.r.t CDT is only minimal. For $RAND$ with node counts of 20, 40 and 60, the time taken to execute the optimization algorithm (GPM) is similar to the time taken for the other methods; however the CDT is high for $RAND$, which leads to a higher TRT when compared to the other methods. For node counts of 80 and 100, the TRT in $RAND$ is comparable to the other methods.

For $I_R = 800m$, the results are very favorable to $DAYG$. As before, $DAYG$ showed lower NC values than $CLIQUDG$ and $RAND$. $DAYG$ also exhibited the smallest CDT when compared to the other two methods, followed by $CLIQUDG$ and $RAND$, in that order. The TRT values were quite similar for $CLIQUDG$ and $RAND$, but were higher than $DAYG$ and $IDEAL$ by at least 0.5 orders of magnitude.

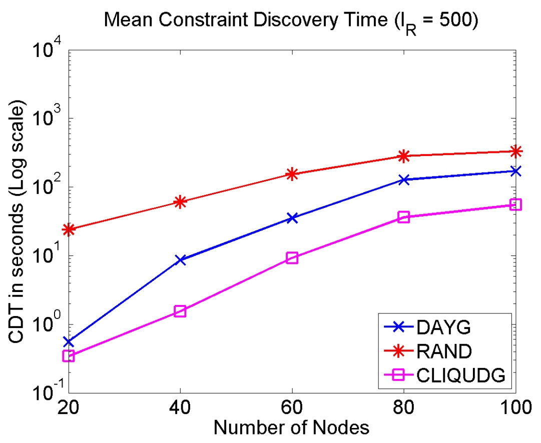
A significant difference with the random topology as compared to the grid topology is that it is very difficult to "verify" the optimal solution for each problem. However, the $IDEAL$, $DAYG$ and $RAND$ methods always converged to the same solution. On the other hand, the solution from $CLIQUDG$ was either equal to (majority of the cases) or less than the optimal solution from $DAYG$. This is consistent with the expected behavior because $CLIQUDG$ might sometimes discover super-maximal cliques, which may result in a feasible, but sub-optimal solution. The sub-optimality is a result of having one or more super-maximal cliques as part of the active set. Table 5.1 shows the average number of super-maximal cliques discovered by $CLIQUDG$ for the random topology. As we can



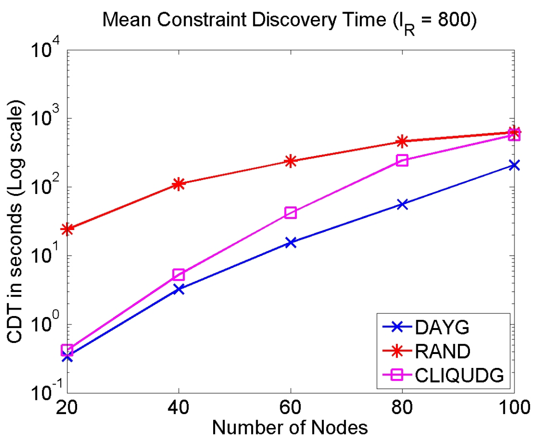
(a) NC for $I_R = 500m$



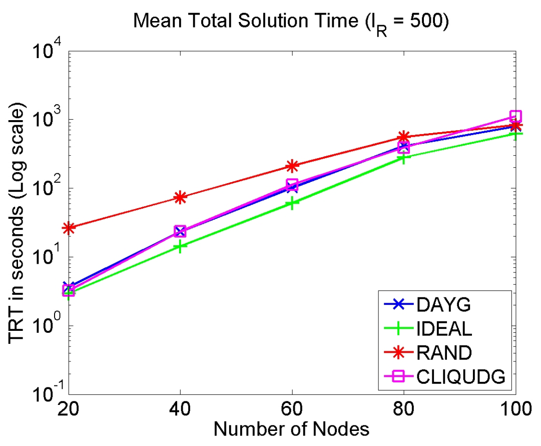
(b) NC for $I_R = 800m$



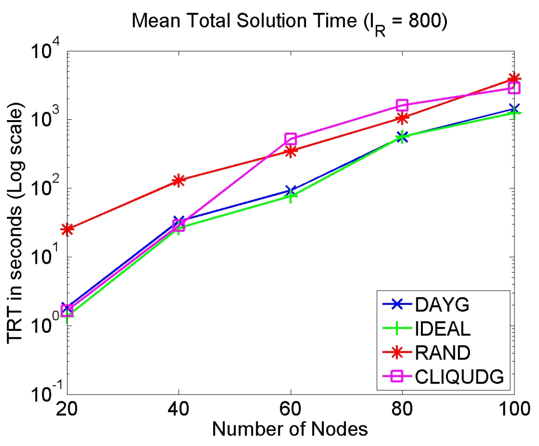
(c) CDT for $I_R = 500m$



(d) CDT for $I_R = 800m$



(e) TRT for $I_R = 500m$



(f) TRT for $I_R = 800m$

Figure 5.3. Performance Plots for Random Topology

see, the number of super-maximal cliques increases as the node (link) density increases. Overall, the impact of super-maximal cliques is only moderate in the cases we investigated. However, this need not be true for other optimization problems that use $CLIQUDG$ for constraint discovery.

Table 5.1. Average Number of Super-Maximal Cliques in $CLIQUDG$ for the Random Topology

	Nodes = 20	*Nodes = 40*	*Nodes = 60*	*Nodes = 80*	*Nodes = 100*
$I_R = 500m$	2.7	38.7	163.4	348.4	437.0
$I_R = 800m$	0.4	45.7	345.2	1480.1	2360.8

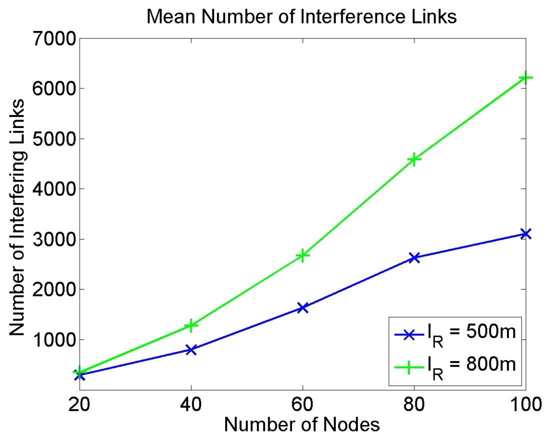
5.4.3 Miscellaneous

Here, we refer to the memory requirement as the amount of memory required to store the inequality constraints, which can be computed as p x q x r, where p is the number of inequality constraints, q is the number of optimization variables, and r is the amount of memory (in bytes) required to store one entry in the inequality constraint matrix. For the single flow $MAXFLOW$ problem (Fig.2.9), the number of variables in the problem is $l + 1$, where l is the sum of number of links in the communication UDG. It has to be noted that changing the interference pattern (from $I_R = 500m$ to $I_R = 800m$) does not impact the number of variables (q) in the optimization formulation. However, it does change the interference UDG and increases the number of nodes that could form a clique, leading to a larger number of unique constraints discovered. Thus the number of variables is purely a function of topology, while the number of constraints is impacted by both the topology and interference pattern. Fig. 5.4a compares the mean number of interference links for the random topologies generated across the two interference patterns. Note that the number of communicating links in each case equals the number of interfering links in the case where $T_R = I_R = 500m$.

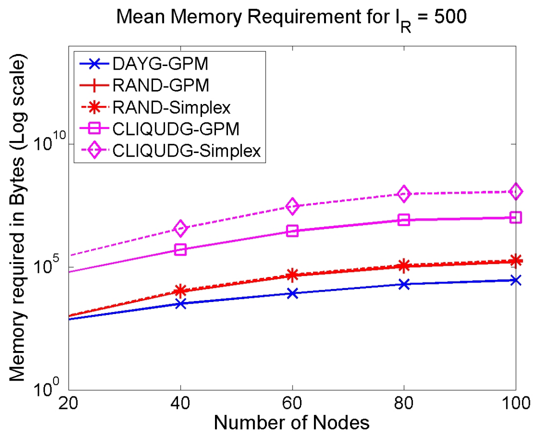
It also has to be noted that the number of variables (q) is also influenced by the optimization algorithm used. The GPM does not require slack or surplus variables, and thus does not artifically inflate the problem size. For example, the Simplex method [Dan51], on the other hand, converts all inequality constraints into equality constraints by adding slack or surplus variables (one variable per constraint). Observe that the number of constraints in network flow problems in MHWNs typically outnumber the number of variables. The Simplex method requires around $r.p.(p+q)$ bytes of memory. Figures 5.4b and 5.4c compare for each method ($DAYG, RAND, CLIQUDG$) the amount of memory required for each random topology considered in the previous section, assuming $r = 1$. In addition, it also compares the amount of memory that would have been required for the Simplex method for $RAND$ and $CLIQUDG$ (computed from above expression). We only show plots pertaining to random topologies in this section, as memory scaling was not an issue with the grid topology experiments.

From Figures 5.4b and 5.4c, the memory savings due to $DAYG$ are abundantly clear, closely followed by $RAND$. Again, note the log scale on the Y-axis. It can also be concluded that the Simplex method may not be a suitable method to solve flow problems for MHWNs while using $CLIQUDG$. The large-scale method uses *lipsol* (linear interior-point solver) that artifically inflates the number of variables to convert an inequality constrained problem to a purely equality constrained one.

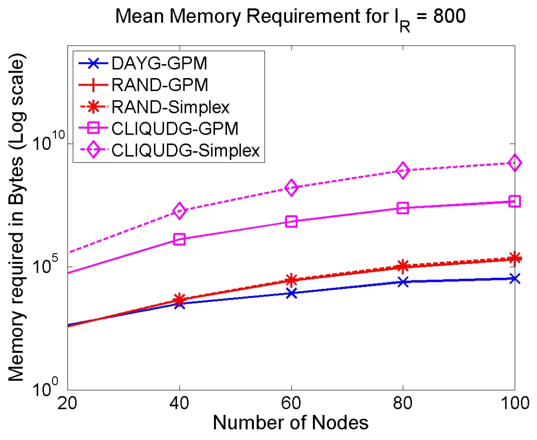
While $RAND$ seems to have performed comparably to $DAYG$ w.r.t memory requirements, it is important to note that this is a direct consequence of using the node-level interference UDG model instead of the link-level model. If the link-level model had been used, then $RAND$ required significantly higher memory



(a) Mean Number of Interfering Links



(b) Memory Requirement ($I_R = 500m$)



(c) Memory Requirement ($I_R = 800m$)

Figure 5.4. Memory Requirements for Random Topology

and run-time to complete execution. This is one of the advantages of using the UDG-based graph model of representing MHWNs.

While the GPM scaled well with problem size, we observed that it was the slowest in terms of execution time when compared to the Simplex method or large-scale optimization methods in MATLAB (for problems that completed execution). Further investigation revealed that the GPM reached the vicinity of the optimal point very quickly, but spent a lot of iterations checking for optimality conditions (deleting constraints from the active set). We suspect that this phenomenon was due to the degeneracy in the optimization variables i.e. a large number of the variables had an optimal value of 0. A value of 0 for any variable x_i will "activate" the inequality corresponding to the non-negative constraints ($x_i \geq 0$). A majority of the "wasted" iterations were perhaps spent adding and deleting these constraints from the active set. Further investigation into this phenomenon and methods of algorithm speed-up will be part of future work.

5.5 Summary

In this chapter, we developed the discover-as-you-go ($DAYG$) method of formulating capacity constraints to solve flow-related optimization problems in MH-WNs. The $DAYG$ method formulated L-capacity constraints *during* the optimization procedure in contrast to conventional methods that require formulating constraints *a priori*. We integrated the $DAYG$ approach into the gradient projection method (GPM) of optimization to obtain the modified gradient projection method ($MGPM$). We then compared the run-time and memory requirements of $DAYG$ with some existing methods ($RAND$ and $CLIQUDG$) by solving the $MAXFLOW$ problem using each of these methods in various grid and random MHWN topologies. For each of these topologies, we varied the number of nodes in the network and interference model parameters and compared the results obtained in each of these cases. Our results showed that in almost all of the cases, $DAYG$ demonstrated superior performance in terms of run-time and memory requirements when compared to the other methods considered.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

One of the major contributions of this dissertation is the introduction of a new concept in networking called the investment function, the development of which will, we believe, serve as a potent tool to devise service differentiation algorithms to address the unique resource allocation and utilization challenges in MHWNs. We demonstrated the effectiveness of this tool by utilizing it to improve multi-hop fairness in multi-hop wireless networks, in addition to improving network utilization efficiency. Through our simulations, we demonstrated two of the many ways that the investment function can be used for better network performance. Our TCP and UDP simulation results with a grid topology indicate that the investment function can provide substantial improvement in throughput and delay fairness properties across multiple hops in addition to substantial reduction in network wastage for UDP flows.

We noted that for optimal investment function definition and service differentiation algorithm design, it might be essential to solve network flow problems in

MHWNs. The shared nature of the wireless medium significantly complicates forumulation of the network-imposed *global* capacity constraints for MHWNs, and thus the optimization problem itself. To overcome this, we have developed an efficient optimization framework for solving network flow problems in MHWNs. As a first step, we modeled the MHWN using a UDG. We observed that under an optimal transmission scheduling scheme, the interference value is given by the chromatic number of the UDG and is very hard to compute. We showed through an empirical study that the clique number, which can be computed in polynomial time for UDGs, can be used as an excellent approximation for the chromatic number (interference) value.

We noted that while the approximation enabled in computing the span M of the optimal schedule in polynomial time, it did not help in formulating the global capacity constraints. Hence, we developed the notion of *Local* or L-capacity constraints that possessed similar properties as the global capacity constraints. We demonstrated that the L-capacity constraints can be forumlated in linear form in polynomial time, and thus can be used in lieu of the global capacity constraints.

Observing that the number of capacity constraints in the network flow problems in MHWNs could be exponentially many, the second part of our study pertains to constructing an efficient constraint formulation method. We leveraged the accurate polynomial-time interference approximation algorithm ($LCONSTR$) to propose the Discover-as-you-go ($DAYG$) approach of discovering the constraints, which discovers constraints only as needed in a dynamic fashion. This is in contrast to existing methods that list a large subset of the constraint set a priori. Through a rigorous set of experiments, we show that our method is computationally very efficient, and requires significantly less memory than existing methods.

This makes our method very scalable with the problem size.

6.2 Future Work

There are many interesting research problems that we would like to solve in the future. First, we would like to formulate the optimization problem for the sample network objective discussed in section 2.3 (improving multi-hop flow fairness and efficiency of network utilization), which is a very challenging task in itself. Based on the results of the optimization algorithm, we would then like to re-visit (and possibly re-design) our sample service differentiation algorithm (section 2.3.1) and the associated investment function definition. We then would like to evaluate the extent of sub-optimality in our algorithm design based on probabilistic and simulation tools. Given the flexibility of the network investment function, we plan to research its various forms and its application to different scenarios such as provision of QoS, distributed fair bandwidth allocation, etc. We would like to devise a distributed scheme that allocates flow-level and node-level bandwidth in a multi-hop wireless network. Another avenue for future work is to study the suitable form of investment function to reduce TCP bias towards 1-hop flows. We also plan to study the fairness-priority tradeoff, by having a larger separation between the user investment factors.

We would like to devise an augmented UDG model that overcomes the current limitations of using a UDG model (discussed in section 3.2.1), and still retains the desirable properties of a UDG model. Currently, the $DAYG$ has been integrated with GPM, and we identified some shortcomings of this approach. We would like to investigate integration of $DAYG$ into other more efficient optimization methods such as the log-barrier method [BV04], so that the optimization is not only memory efficient, but also much faster than $MGPM$. We would also like to compare our method with existing non-differentiable methods of optimization (NDO) [EGV01], and explore methods of developing highly efficient MHWN-specific algorithms for a variety of generalized network flow problems in MHWNs (such as Ford-Fulkerson [LD56] for $MAXFLOW$ in wired networks).

Bibliography

[80206] IEEE 802.15.4-2006 Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf, 2006.

[80207] IEEE 802.11,2007 Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. http://standards.ieee.org/getieee802/download/802.11-2007.pdf, 2007.

[All] D. Allen. “Hidden Terminal Problems in Wireless LAN’s”. IEEE 802.11 Working Group paper 802.11/93-xx.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, USA, 1993.

[AWW05] Ian F. Akyildiz, Xudong Wang, and Weilin Wangb. “Wireless mesh networks: a survey”. *Computer Networks*, 47(4):445–487, March 2005.

[BBC^{+}] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. IETF RFC 2475.

[BCD90] B.N.Clark, C.J.Colbourn, and D.S.Johnson. "Unit Disk Graphs". *Discrete Mathematics*, 86:165–177, 1990.

[BCS] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. IETF RFC 1633.

[BDSZ94] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. "MACAW: a media access protocol for wireless LAN's". *SIGCOMM Comput. Commun. Rev.*, 24(4):212–225, 1994.

[BJ08] Osama Bazan and Muhammad Jaseemuddin. "Multi-commodity Flow Problem for Multi-hop Wireless Networks with Realistic Smart Antenna Model". In *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, volume 4982 of *LNCS*, pages 922–929. Springer, Berlin/Heidelberg, 2008.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, March 2004.

[BZB^{+}] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP). IETF RFC 2205.

[CdGB07] Tom Coenen, Maurits de Graaf, and Richard J. Boucherie1. "An Upper Bound on Multi-hop Wireless Network Performance". In *Managing Traffic Performance in Converged Networks*, volume 4516 of *LNCS*, pages 335–347. Springer, Berlin/Heidelberg, 2007.

[cli] Cliquer - routines for clique searching. http://users.tkk.fi/ pat/cliquer.html.

[Dan51] G. B Dantzig. “Maximization of a Linear Function of Variables Subject to Linear Inequalities”. In T.C.Koopmans, editor, *Activity Analysis of Production and Allocation*, volume 13 of *Cowles Commission Monograph*, pages 339–347. John Wiley, New York, 1951.

[dim] DIMACS: Center for Discrete Mathematics and Theoretical Computer Science. http://prolland.free.fr/works/research/dsat/dimacs.html.

[DLV04] Jing Deng, Ben Liang, and P.K. Varshney. “Tuning the carrier sensing range of IEEE 802.11 MAC”. *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, 5:2987–2991 Vol.5, Nov.-3 Dec. 2004.

[dsa] Graph Coloring by DSATUR. http://mat.gsia.cmu.edu/COLOR/color.html.

[EF06] T. Erlebach and J. Fiala. “Independence and coloring problems on intersection graphs of disks”. In E.Bampis, K.Jansen, and C.Kenyon, editors, *Efficient Approximation and Online Algorithms*, volume 3484 of *LNCS*, pages 135–155. Springer, Heidelberg, 2006.

[EGV01] S. Elhedhli, J.-L. Goffin, and J.-P. Vial. “Nondifferentiable Optimization: Introduction, Applications and Algorithms”. *Encyclopedia on Optimization*, 2001.

[EK72] Jack Edmonds and Richard M. Karp. “Theoretical improvements in algorithmic efficiency for network flow problems”. *Journal of the ACM*, 19(2):248–264, 1972.

[FZL$^+$03] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. “The impact of multihop wireless channel on TCP throughput and loss”. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, volume 3, pages 1744–1753, 2003.

[GK00] P. Gupta and P.R. Kumar. “The capacity of wireless networks”. *IEEE Transactions on Information Theory*, 46(2):388–404, march 2000.

[GM01] Gerke.S and McDiarmid.C. “Graph Imperfection”. *Journal of Combinatorial Theory*, 83(1):58–78, September 2001.

[GMW07] Rajarshi Gupta, John Musacchio, and Jean Walrand. “Sufficient rate constraints for QoS flows in ad-hoc networks”. *Ad Hoc Networks*, 5(4):429–443, 2007.

[GSW98] A. Graf, M. Stumpf, and G. Weienfels. “On Coloring Unit Disk Graphs”. *Algorithmica*, 20(3):277–293, 1998.

[GTB99] Mario Gerla, Ken Tang, and Rajive Bagrodia. “TCP Performance in Wireless Multi-hop Networks”. In *WMCSA ’99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, Washington, DC, USA, 1999. IEEE Computer Society.

[gum] *GumStix*. http://gumstix.com/.

[GW04] R. Gupta and J. Walrand. “Approximating maximal cliques in ad-hoc networks”. In *Proceedings of 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 365–369, Barcelona, Spain, September 2004.

[GWG05] R. Gupta, J. Walrand, and O. Goldschmidt. “Maximal Cliques in Unit Disk Graphs: Polynomial Approximation”. In *Proceedings of International Network Optimization Conference*, Lisbon, Portugal, March 2005.

[Hal80] W.K Hale. “Frequency assignment: theory and applications”. *Proceedings of IEEE*, 68:1497–1513, 1980.

[JB01] Ramirez-Alfonsin J. and Reed B. *Perfect Graphs.* J.H. Wiley and Sons, Chichester U.K, 2001.

[JCH84] R. Jain, D. Chiu, and W. Hawe. “A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems”, 1984.

[JPPQ03] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. “Impact of interference on multi-hop wireless network performance”. In *Proceedings of Int’l Conf. Mobile Computing and Networking (Mobicom)*, pages 66–80, San Diego, CA, 2003.

[Kar39] W. Karush. “Minima of Functions of Several Variables with Inequalities as Side Constraints”. Master’s thesis, Dept. of Mathematics, University of Chicago, Chicago, IL, 1939.

[KMPS05] V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. “Algorithmic aspects of capacity in wireless networks”. In *SIGMETRICS ’05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and model-*

ing of computer systems, pages 133–144, New York, NY, USA, 2005. ACM.

[KMS] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals (RFC 4919). IETF RFC 4919.

[KN05] Murali Kodialam and Thyaga Nandagopal. "Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels". *IEEE/ACM Transactions on Networking*, 13(4):868–880, 2005.

[KT51] H. W. Kuhn and A. W. Tucker. "Nonlinear programming". In *Proceedings of 2nd Berkeley Symposium*, pages 481–492, Berkeley, 1951. University of California Press.

[LAZC00] Seoung-Bum Lee, Gahng-Seop Ahn, Xiaowei Zhang, and Andrew T. Campbell. "INSIGNIA: An IP-Based Quality of Service Framework for Mobile ad Hoc Networks". *Journal of Parallel and Distributed Computing*, 60(4):374–406, April 2000.

[LBD$^+$01] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. "capacity of ad hoc wireless networks". In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pages 61–69, Rome, Italy, July 2001.

[LBS99] Songwu Lu, Vaduvur Bharghavan, and R. Srikant. "Fair scheduling in wireless packet networks". *IEEE/ACM Transactions on Networking*, 7(4):473–489, 1999.

[LC05] Xuefei Li and L. Cuthbert. “Multipath QoS routing of supporting DiffServ in mobile ad hoc networks”. *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on*, pages 308–313, May 2005.

[LD56] L.R.Ford and D.R.Fulkerson. “Maximal flow through a network”. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[Lei79] F.T. Leighton. “A graph coloring algorithm for large scheduling problems”. *Journal of Research of the National Bureau of Standards*, 84:489–506, 1979.

[Li05] Baochun Li. “End-to-End Fair Bandwidth Allocation in Multi-hop Wireless Ad Hoc Networks”. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS 2005)*, pages 471–480, Columbia, Ohio, June 2005.

[LQZ$^+$07] Yi Li, Lili Qiu, Yin Zhang, Ratul Mahajan, Zifei Zhong, Gaurav Deshpande, and Eric Rozner. “Effects of interference on throughput of wireless mesh networks: Pathologies and a preliminary solution.”. In *Proceedings of Sixth Workshop on Hot Topics in Networks (HotNets-VI)*, Atlanta, GA, November 2007.

[Lue84] David G. Luenberger. *Linear and NonLinear Programming.* Addison Wesley, Reading, MA, 1984.

[man] *Mobile Ad hoc Networking (MANET) Charter.* http://www.ietf.org/html.charters/manet-charter.html.

[MM65] J.W. Moon and L. Moser. “On cliques in graphs”. *Israel Journal of Mathematics*, 3:23–28, 1965.

[mot] *MICA MOTES from Crossbow Technologies.* http://www.xbow.com/Products/productdetails.aspx?sid=164.

[MP06] Pradeepkumar Mani and David W Petr. “Investment Function: Enhanced Fairness and Performance in Multi-hop Wireless Networks”. *Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on*, pages 721–728, October 2006.

[MP07] Pradeep Mani and David Petr. “Clique number vs. chromatic number in wireless interference graphs: Simulation results”. *IEEE Communication Letters*, 11(7):592–594, July 2007.

[NKGB00] Thyagarajan Nandagopal, Tae-Eun Kim, Xia Gao, and Vaduvur Bharghavan. “Achieving MAC layer fairness in wireless packet networks”. In *MobiCom ’00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 87–98, New York, NY, USA, 2000. ACM.

[ns2] *ns-2 simulator.* http://nsnam.isi.edu/nsnam/index.php/User_Information.

[Pro00] John G. Proakis. *Digital Communications.* McGraw-Hill, USA, 4 edition, August 2000.

[QM03] Y. Qiu and P. Marbach. "Bandwidth allocation in ad hoc networks: a price-based approach". *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2:797–807, March - April 2003.

[Rap01] Theodore Rappaport. *Wireless Communications:Principles and Practice* . Prentice Hall, USA, 2 edition, 2001.

[RDS$^+$07] Ashish Raniwala, Pradipta De, Srikant Sharma, Rupa Krishnan, and Tzi cker Chiueh. "End-to-End Flow Fairness over IEEE 802.11-based Wireless Mesh Networks". In *Proc. of IEEE Infocom Mini-Sympsium on 802.11 Wireless LANs*, May 2007.

[Ros60] J.B. Rosen. "The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints". *Journal of the Society for Industrial and Applied Mathematics*, 8(1):181–217, March 1960.

[RS05] Ananth Rao and Ion Stoica. "An overlay MAC layer for 802.11 networks". In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 135–148, New York, NY, USA, 2005. ACM.

[SAHS05] Karthikeyan Sundaresan, Vaidyanathan Anantharaman, Hung-Yun Hsieh, and Raghupathy Sivakumar. "ATP: A Reliable Transport Protocol for Ad Hoc Networks". *IEEE Transactions on Mobile Computing*, 4(6):588–603, 2005.

[SDS97] S.Hurley, D.H.Smith, and S.U.Thiel. "FASoft: A system for discrete channel frequency assignment". *Radio Science*, 32(5):1921–1939, September 1997.

[SHS04] Karthikeyan Sundaresan, Hung-Yun Hsieh, and Raghupathy Sivakumar. "IEEE 802.11 over multi-hop wireless networks: problems and new perspectives". *Ad Hoc Networks*, 2(2):109–132, April 2004.

[Str92] Timothy Strayer. *"Function-Driven Scheduling: A General Framework for Expression and Analysis of Scheduling"*. PhD thesis, Dept. of Comp. Sci, University of Virginia, Charlottesville, Virginia, 1992.

[sun] *Sun Small Programmable Object Technology (SunSpots).* http://www.sunspotworld.com/.

[vRSWZ05] Pascal von Rickenbach, Stefan Schmid, Roger Wattenhofer, and Aaron Zollinger. "A Robust Interference Model for Wireless Ad-Hoc Networks". In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 12*, page 239.1, Washington, DC, USA, 2005. IEEE Computer Society.

[Wer85] D. De Werra. "An introduction to timetabling". *European Journal of Operations Research*, 19:151–162, 1985.

[XLN06] Yuan Xue, Baochun Li, and Klara Nahrstedt. "Optimal resource allocation in wireless ad hoc networks: a price-based approach". *Mobile Computing, IEEE Transactions on*, 5(4):347–364, April 2006.

[XPMS01] G. Xylomenos, G. Polyzos, P. Mahonen, and M. Saaranen. “TCP Performance Issues over Wireless Links”. *IEEE Communications Magazine*, 39(4):52–58, 2001.

[XS01] S. Xu and T. Saadawi. “Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?”. *Communications Magazine, IEEE*, 39(6):130–137, 2001.

[XSLC00] Hannan Xiao, W.K.G. Seah, A. Lo, and K.C. Chua. “A flexible quality of service model for mobile ad-hoc networks”. *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, 1:445–449, 2000.

[YPK03] Su Yi, Yong Pei, and Shivkumar Kalyanaraman. “On the capacity improvement of ad hoc wireless networks using directional antennas”. In *MobiHoc ’03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 108–116, New York, NY, USA, 2003. ACM.

[YYS03] Fengji Ye, Su Yi, and B. Sikdar. “Improving spatial reuse of IEEE 802.11 based ad hoc networks”. *Global Telecommunications Conference, 2003. GLOBECOM ’03. IEEE*, 2:1013–1017 Vol.2, Dec. 2003.

[zig] *ZigBee Alliance.* http://www.zigbee.org/en/index.asp.