Resource Allocation Problems in G.fast and Enterprise Wi-Fi Networks

Student Name: Ankita Raj

Roll Number: MT14077

July 12, 2016

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee
Dr. Pravesh Biyani (Chair)
Dr. Sanjit Krishnan Kaul
Dr. Ketan Rajawat

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Electronics and Communication Engineering,
with specialization in Communication and Signal Processing

Keywords: Resource allocation, G.fast, Wi-Fi, weighted A^* algorithm, combinatorial optimization.

Certificate

This is to certify that the thesis titled **"Resource allocation problems in G.fast and Enterprise WiFi Networks"** submitted by **Ankita Raj** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *Electronics and Communication & Engineering* is a record of the bona fide work carried out by her under my guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. Pravesh Biyani
Indraprastha Institute of Information Technology, New Delhi

Abstract

This thesis addresses resource allocation problems in two different domains: G.fast and Wi-Fi.

One of the key challenges in G.fast is to minimize power consumption at the distribution points. G.fast standards define Discontinuous Operation modes that provide avenues for power reduction by allowing intermittent transmission of users along time slots. We formulate power efficiency problem as a user-slot assignment problem, where we schedule users to time slots during discontinuous operation such that the total power consumption is minimized. Since the user-slot assignment is an NP complete problem, a weighted-A^* based algorithm has been proposed that achieves reasonable performance with limited computational resources. The proposed framework also explores the inter-relationship between precoding, bit-loading and user-slot assignment, and their combined impact on power consumption. The assignment algorithm combined ensures that a simple precoding scheme compares favorably with more complex precoding schemes like discontinuous vectoring, in terms of energy efficiency. We also observe that applying our user-slot assignment on top of discontinuous vectoring substantially enhances its energy efficiency as well.

A similar resource allocation problem exists for the maximization of throughput in 802.11 wireless LAN's. A prior study proposed that by allowing links to transmit data simultaneously by piggybacking on each other's data transmission opportunities, network throughput can be increased[1]. Further, it formulated a network partition problem for deciding which links in the network must transmit simultaneously to maximize throughput. The problem is NP hard in nature, and so far only heuristic methods have been proposed to solve it. In this thesis, we re-frame the network partitioning problem as a resource allocation problem similar to the one for power minimization in G.fast, with a few additional constraints. We use an analogous weighted-A^* based framework for maximization problems to solve the same. We show that our algorithm achieves large throughput gains, with provable bounds on the quality of the solution obtained.

[1]Mridula Singh, Sanjit Kaul, and Pravesh Biyani."On large throughputs in high density enterprise wireless LAN (s)." Global Communications Conference (GLOBECOM), 2014 IEEE.

Acknowledgments

I would like to express my deepest gratitude to my advisor Dr. Pravesh Biyani for providing invaluable guidance and encouragement, and for his patience all the way.

I sincerely thank Dr. Sandip Aine for his valuable inputs for the A^* algorithm. I would like to thank Dr. Sanjit Krishnan Kaul for his guidance in the resource allocation problem in Wi-Fi, and for providing great insights into the prior work in the area. My sincere thanks goes to Dr. Surendra Prasad (IIT Delhi) for his valuable suggestions.

I wish to express my gratitude to all my professors for the wonderful courses they taught, and to IIITD in general for providing such a great environment to work. A special thanks to my friends, especially Arpita, for helpful discussions and critical reviews on my work. These are the people who made my stay at IIITD an enjoyable one, and kept me going.

Finally and most importantly, I would like to thank my parents and my brother for being the most vital source of encouragement and positive energy in my life.

Contents

I Resource allocation in G.fast **1**

1 Introduction **2**

1.1 Power reduction challenge in G.fast 2

1.2 Existing Work 3

1.3 Our contributions/Insights 4

1.4 Notations 5

2 System Model **6**

2.1 G.fast architecture 6

2.2 Channel model and rate calculations 6

2.3 Vectoring model 8

2.4 Energy consumption model 9

3 Discontinuous Problem Formulation **10**

4 Algorithm **11**

4.1 Motivation 11

4.2 The A^* algorithm 11

4.2.1 Weighted A^* algorithm 12

4.3 Admissible heuristic for A^* 13

4.4 An example 15

4.5 Reducing the computational complexity 16

5 Simulation Setup **17**

5.1 Topology 17

5.2 Traffic patterns 17

5.3 Bit-loading and attainable data rates 18

6 Results **20**

6.1 Comparison of Energy efficiency 20

6.2 Time complexity of the weighted A^* algorithm 24

II Resource allocation in WiFi **25**

7 Introduction **26**

7.1 Network model and throughput calculation 26

7.2 Optimization Problem 27

8 Proposed approach **28**

8.1 A^* algorithm 28

8.2 Admissible heuristic 30

9 Results **31**

9.1 Simulation Setup 31

9.2 Results 31

10 Conclusion **33**

Appendix A Proof of Theorem 1 **34**

List of Abbreviations

AP	Access Point
AWGN	Additive White Gaussian Noise
BSS	Begin Sharing in Space
CTS	Clear To Send
DCF	Distributed Coordination Function
DMT	Discrete Multitone
DO	Discontinuous Operation
DP	Dynamic Programming
DPU	Distribution Point Unit
DRA	Dynamic Resource Allocation
DSL	Digital Subscriber Line
DV	Discontinuous Vectoring
FTTdp	Fiber to the Distribution Point
GCBP	Bin packing with general cost structures
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
LAN	Local Area Network
NP	Nondeterministic Polynomial time
OIP	Optimal Integer Partition
PSD	Power Spectral Density
RMC	Robust Management Channel
ROIP	Reevaluate Optimal Integer Partition
RTS	Request To Send
SINR	Signal to Interference plus Noise Ratio
SNR	Signal to Noise Ratio
SS	Share-in-Space
TDD	Time Division Duplexing
TP	Traffic Pattern
TxOP	Transmit Opportunity
WLAN	Wireless Local Area Network

Part I

Resource allocation in G.fast

Chapter 1

Introduction

1.1 Power reduction challenge in G.fast

Over the past several years, demand for higher data rates has compelled broadband access technology to migrate from traditional copper telephony wire based transmission to optical fiber based networks. But in most practical scenarios, it is economically infeasible to provide fiber connections to each subscriber. This has led to the emergence of hybrid copper-fiber access networks that deploy fiber for most part of the loop, except in the highly branched topologies very close to the end-users, where the existing copper assets are used [1]. Such hybrid networks are aimed to serve as the transition from copper-based to pure fiber-based networks.

G.fast is one such emerging broadband standard approved by the International Telecommunication Union (ITU), tailored specifically for the Fiber to the Distribution Point (FTTdp) deployment model, wherein optical fiber from the central office terminates in a distribution point unit (DPU) close to the end-users, and the remaining few hundred meters are covered by existing copper infrastructure. G.fast aims to provide aggregate data rates of up to 1 Gbps over short copper loops.

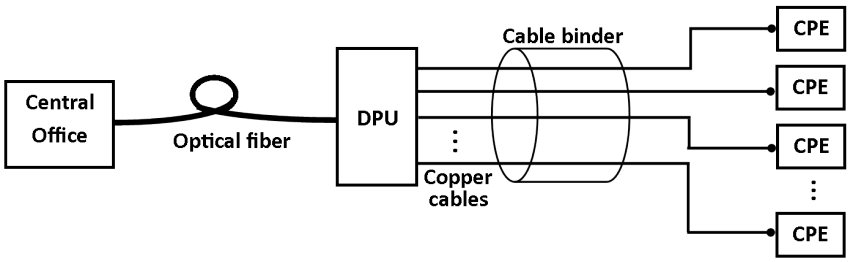


Figure 1.1: G.fast FTTdp architecture

To avoid the need for a local power supply, the distribution point unit is reverse powered from the customer premises over the existing copper wires. There is hence a power bottleneck at the DPU and consequently the need to minimize the power consumption. In general, to enable energy efficiency, the G.fast standards define what is known as discontinuous operation (DO), which

are specifically designed power saving modes that allow intermittent transmissions for different users. Note that the G.fast transceivers use synchronized time division duplexing (TDD) and each TDD frame comprises of dedicated symbol positions (or time slots) for upstream and downstream data transmission. In the DO mode, transceivers can be disabled in the time slots they do not need to transmit data, thereby reducing the power consumption.

When users in the same binder transmit simultaneously, electromagnetic coupling between the neighbouring twisted pairs causes interference, known as crosstalk. Traditional DSL systems employ a technique known as vectoring for mitigation of crosstalk, which involves joint processing of the transmit signals on all the twisted pairs within a binder so that crosstalk is canceled [11], [7]. In G.fast, however, owing to discontinuous operation, not all users are active in all time slots. This context presents a new challenge in designing a precoder for crosstalk mitigation in G.fast. Precoding all users leads to idle symbols being transmitted on the disabled lines as well, thereby sacrificing the power saving potential of DO (as explained later). On the other hand, the idea of precoding only the set of active users in each time slot is practically infeasible since it calls for recomputing the precoder in every time slot. Hence, there is the need for an efficient vectoring strategy suited to Discontinuous Operation.

1.2 Existing Work

Maes and Nuzman in [14] were the first to discuss precoder design for DO in G.fast. Authors describe two methods that allow turning off analog components during discontinuous operation while employing a precoder. The first method, *muted vectoring*, is to simply mute the output of the precoder on the discontinued lines. The second method called *discontinuous vectoring (DV)* relies on controlling the structure of the data that should be precoded in order to satisfy no transmission requirement for users in low-power mode without disturbing the precoder. Although both methods result in residual crosstalk for the active users, DV causes relatively low residual crosstalk and hence exhibits reasonable power saving potential. The precoder (for a given tone) in DV does not change with every time slot, but the input to the precoder still needs to be recomputed at every slot as the set of active users changes, which puts an extra computational burden on the system.

While the method proposed in [14] does achieve reasonable energy efficiency, there are other aspects to the problem that may provide further scope for power reduction. In G.fast, the SNR of users changes in every time slot as the set of active users changes, however the bit-loadings (number of bits allocated to each subcarrier) of active users in [14] are determined in accordance with the maximum residual noise (i.e. the minimum SNR) over all user sets, and thus fixed for all time slots. This amounts to achieving lower than the maximum attainable data rates. In contrast, the G.fast standard contains provision to modify the active bit loading tables in every time slot. This is communicated via the Robust Management Channel (RMC) symbol contained in each TDD frame, which is responsible for fast reconfiguration and physical layer management [21]. The mechanism is described in section 9.6 of ITU-T Recommendation

G.9701 [2]. This makes it possible to allow the bit-rates to change in every time slot in accordance with the SNR, assuming that the bit loadings for all slots are available at the beginning of each TDD frame.

1.3 Our contributions/Insights

In this work, we take a relook at the energy efficiency problem for the discontinuous operations in G.fast. We essentially treat this problem as an optimization problem where in the users are to be allocated time slots in a TDD frame with an objective to minimise the overall power consumption while satisfying the constraints on the data rate of all the users. The rate of a user depends on the SNR of that user in the presence of others and the precoding strategy that is applied in order to mitigate the crosstalk. We look at various possible options to precode interfering users (including the DV by [14]) to calculate the data rate for a given set of users, as well as at the possibility of varying bit-loading for users across time-slots as suggested by the G.fast standard. Allowing the bit-loadings to change in every time slot in accordance with the changing SNR offers the possibility of achieving higher rates as compared to assigning constant bit-loadings. In summary, we frame the power efficiency problem in DO as a time slot to user assignment problem of minimizing the total number of active symbol positions occupied by all users over a TDD frame while achieving a minimum target data rate for each user. We also explore the inter-relationships between precoding, bit-loading and user-slot assignment, and their combined impact on the power consumption.

The user-slot assignment problem is combinatorial in nature and can be proved to be an NP hard problem. We propose an A^* based algorithm to obtain an optimal (or bounded sub-optimal) user-slot assignment in an efficient manner. A^* requires an admissible heuristic that helps in pruning the search space without losing optimal solutions. We obtain an admissible heuristic by solving a relaxed version of the original problem using dynamic programming, which provides a provable lower bound to the optimal cost. While A^* with an admissible heuristic is both optimal (i.e., it guarantees optimal termination) and optimally efficient (i.e., among all such search algorithms that traverse the same search space, A^* is guaranteed to make the smallest possible number of search steps), sometimes the memory/time requirement of A^* becomes infeasible for practical applications. For practical implementation and simulations we use weighted A^* [17], which converts A^* to a (provably) bounded suboptimal algorithm and takes significantly less computational time and memory.

The proposed user-slot assignment approach can be employed in conjunction with any precoding scheme. We consider a simple scheme whereby precoding is done only in those time slots where all the users are transmitting together (normal mode), whereas for any other user-combination, no precoder is employed (during DO). Simulations are performed on field measurement data under practical traffic scenarios to evaluate the analog power consumption at the DPU for this precoding scheme. Results show that the simple precoding method combined with user-slot allocation slightly outperforms the discontinuous vectoring based precoding strategy in [14]. In

fact, performing user-slot assignment also improves the performance of the DV (assuming bit-loading also changes with time-slots) and in many cases this outperforms the suggested simple precoding strategy. Thus, using the user slot assignment algorithm provides more options for the G.fast system designers while maintaining the power efficiency.

1.4 Notations

We use the following notations in the subsequent chapters. Bold upper- and lower-case letters denote matrices and column vectors. The $(i, j)^{th}$ element of matrix $\mathbf{A}$ is represented by $a_{i,j} = [\mathbf{A}]_{i,j}$.

Chapter 2

System Model

2.1 G.fast architecture

The G.fast architecture consists of optical fiber terminating in a Distribution Point Unit (DPU) very close to the customer premises. Each DPU is connected to a limited number of end-users via existing copper wires [13]. Each subscriber is served via a dedicated twisted pair, and many such copper pairs are grouped together in a bundle. G.fast standards define two profiles with in-band spectral usage of up to 106 MHz and 212 MHz respectively and uses DMT multi-carrier modulation. The 106 MHz profile consists of 2048 sub-carriers with a spacing of 51.75 kHz [6]. Because each subscriber uses the entire spectrum, subscribers sharing the same binder are subject to crosstalk, which is considerably higher for the higher frequencies.

Unlike traditional DSL which uses Frequency Division Duplexing, G.fast uses Time Division Duplexing (TDD). A G.fast TDD frame (36 time slots) consists of DMT symbol positions allocated to downstream data followed by upstream data. The set of symbol positions in a TDD frame that may be used for data transmission is known as a Transmit Opportunity (TXOP). A Dynamic Resource Allocation (DRA) engine controls the number of symbol positions allocated to each transmit direction [14]. Both the downstream and upstream data blocks are further divided into normal and discontinuous intervals. In the normal mode, all the users are active simultaneously, whereas in the discontinuous mode, some of the users are active while some may stop transmitting to save power (refer Figure2.1) [21].

2.2 Channel model and rate calculations

Consider a system comprising N lines grouped together in a binder. Each subscriber uses a set of subcarriers $k = 1, \dots K$ for transmission over the crosstalk channel $\mathbf{H}^{(k)} \in \mathbb{C}_{N \times N}$. The system model for the k^{th} tone is given by:

$$\mathbf{y}^{(k)} = \mathbf{H}^{(k)}\mathbf{u}^{(k)} + \mathbf{n}^{(k)} \tag{2.1}$$

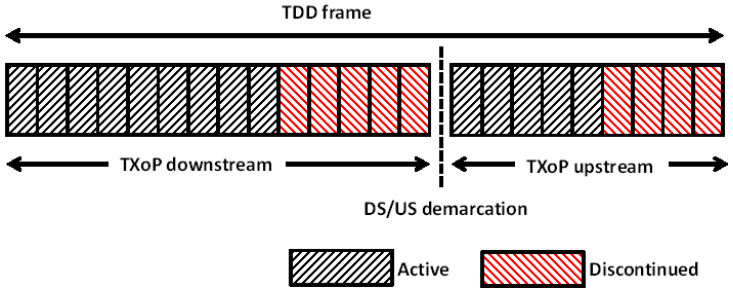


Figure 2.1: A TDD frame comprises of downstream and upstream symbol positions. In each direction, the line remains active for some symbol positions and may go into discontinued mode when there is no data to transmit.

where $\mathbf{u}^{(k)}$, $\mathbf{y}^{(k)}$ and $\mathbf{n}^{(k)}$ denote the users' DMT signal, received signal and additive noise respectively for subcarrier k. Channel $\mathbf{H}^{(k)}$ for each subcarrier k is known at the transmitter.

In a given time slot, the SNR for line l at subcarrier frequency k is calculated as [25]:

$$\gamma_l^{(k)} = \frac{P_l^{(k)} \left| h_{l,l}^{(k)} \right|^2}{\sum_{m \in \mathbb{M}, m \neq l} P_m^{(k)} \left| h_{l,m}^{(k)} \right|^2 + N_l^{(k)}} \tag{2.2}$$

where $P_l^{(k)}$ is the transmit power for line l, $N_l^{(k)}$ is the noise power for line l, $h_{l,m}^{(k)}$ is the crosstalk channel from line m to line l (i.e. $h_{l,m}^{(k)} = [\mathbf{H}^{(k)}]_{l,m}$), and $\mathbb{M}$ denotes the set of active users in a given time slot. The SNR of users varies from one time slot to another as the set of active users in the slots changes.

Given the SNR, the number of bits that can be reliably transmitted in one time slot by line l at tone k is computed as follows [8]:

$$b_l^{(k)} \leq \min \left(\lfloor \log_2 \left(1 + \frac{\gamma_l^{(k)}}{\Gamma} \right) \rfloor, b_{max} \right) \tag{2.3}$$

where Γ is the SNR gap computed as in [8] and b_{max} is the maximum bit-loading per constellation. In practice, bit-loading for line l at tone k may either be assigned as per eq (2.3), in which case it would change with every time slot, or it could be assigned a fixed value that does not exceed the maximum limit determined by eq (2.3) for any user combination.

The total rate for line l, aggregated over upstream and downstream, is determined as [25]:

$$R_l = \eta f_c \sum_{k=1}^{K} b_l^{(k)} \tag{2.4}$$

where η is the efficiency (includes transmission efficiency and coding overhead [13]), f_c is the sub-carrier spacing (51.75 kHz) and $b_l^{(k)}$ is the bit-loading at sub-carrier k, obtained directly using eq (2.3), or a lower value as discussed above. If bit-loadings are allowed to change from one time slot to another, the corresponding rate also varies accordingly.

2.3 Vectoring model

Since different users in a binder cause considerable crosstalk to each other, it becomes essential to precode the users in order to cancel the crosstalk among them, so that higher data rates can be achieved. There have been studies analysing the performance of both linear and non-linear precoding schemes with G.fast [22], [15]. In this paper, we have used the diagonalizing precoder with column-norm scaling described in [14] to mitigate crosstalk while satisfying the PSD constraints. When a precoder is used, the system model of eq (2.1) takes the following form (omitting superscript k for convenience):

$$\mathbf{y} = \mathbf{F}(\mathbf{Hx} + \mathbf{n}) = \mathbf{F}(\mathbf{HPu} + \mathbf{n}) \tag{2.5}$$

where $\mathbf{u}$, $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{n}$ denote the users' DMT signal, transmitted signal, received signal and additive noise respectively. $\mathbf{H}$ is the channel matrix, $\mathbf{P}$ is the precoding matrix, and $\mathbf{F}$ is the diagonal equalizer at the receiver to compensate channel distortion. The precoder can be decomposed as $\mathbf{P} = \mathbf{P'S}$, where $\mathbf{P'} = \mathbf{H^{-1}}$, and $\mathbf{S}$ is the scaling matrix that compensates for the power increase due to the precoder coefficients, computed as in [14].

With a precoder applied, crosstalk among the users is reduced, and so the data transmission rate of the users increases. The SNR for line l at tone k is calculated as in eq (2.2), with $h_{l,m}^{(k)} = [\mathbf{H^{(k)}} \times \mathbf{P^{(k)}}]_{l,m}$. Bit-loading corresponding to the updated SNR is calculated either for each time slot using eq (2.3), or fixed a-priori considering the worst-case SNR.

The basic precoder structure described above can be adapted to different precoding strategies suited to discontinuous operation. As mentioned earlier, the precoding strategy and bit-loading strategy are both important factors in determining the power consumption at the DPU besides the user-slot assignment. Based on these considerations, we analyze the following precoding-cum-bit-loading strategies in our work:

- *Precode-N:* This method involves precoding only when all users in the binder are simultaneously active. Here we allow the bit-rates to be updated for every user-combination, and only two precoder structures are required —one when all users are active, and another (no precoder) for any other user combination.

- *DV with variable bit-rates:* We employ the Discontinuous vectoring technique proposed by Maes and Nuzman in [14] with the modification that, the bit-rates, instead of being fixed according to the maximum residual noise level, are allowed to vary with the user-combination. Although the precoder structure remains the same throughout, the virtual input for the discontinued lines needs to be updated as the set of discontinued lines changes.

- *DV with frame-wise fixed bit-rates:* It is similar to the previous scheme, except for the fact that the bit loadings of all users are fixed for every TDD frame by estimating the minimum bit-rates required to achieve the target rates for the current time frame. The virtual input on the discontinued lines still needs to be updated for every time slot.

2.4 Energy consumption model

The energy consumption model described in [14] is used to evaluate the power consumption of the analog functionality under different modes of operation. In the active state, the analog power consumption per line for a single time slot is modeled with $P_f = 500$ mW. In the discontinuous mode, depending upon whether the discontinued lines transmit idle symbols or not, power consumption of the discontinued users may be either modeled as full $P_f = 500$ mW, or as reduced $P_r = 150$ mW.

Chapter 3

Discontinuous Problem Formulation

The problem of scheduling users to time slots in DO to minimize the power consumption requires as its inputs the data rate requirements of the users, and the bit-rates for different possible active/discontinued user combinations, which in turn depend on crosstalk and the precoding strategy applied. The energy efficient user-slot assignment problem formulation is the following.

Problem 1: Let there be a total T slots and N users. Let Π denote a set of precoding strategies, and $p \in \Pi$. Define a set $S \subset [1, N]$ and $i \in S$. Let $R_i(S, p)$ denote the rate of user i (over all subcarriers) when it shares a time-slot with user set S while adopting a precoding strategy p. Let the matrix $\mathbf{A}$ be $N \times T$ association matrix, such that when user i transmits in slot j, $a_{i,j} = [\mathbf{A}]_{i,j} = 1$, otherwise $a_{i,j} = 0$. Let the total target rate for user i be R_i^{target}. We have the following user-slot assignment problem.

$$\begin{aligned} &\text{Minimise} \quad \sum_{i=1}^{N} \sum_{j=1}^{T} a_{i,j} \\ &\text{Subject to} \quad R_i^{total} \geq R_i^{target} \quad \forall i \end{aligned} \tag{3.1}$$

where $R_i^{total} = \sum_{j:S_j \ni i} R_i(S_j, p)$ is the total rate obtained for user i over all T slots (i.e. the average bits transmitted per second by the user in the corresponding TDD frame), when the set of users S_j determined by the association matrix $\mathbf{A}$ is active in time slot j. The objective is to find an optimal time slot allocation pattern that minimizes the total power consumption (which can be mapped to the total number of 1's in $\mathbf{A}$), while satisfying each user's rate constraint. For each time slot, there are 2^N ways of allocating it to $0, \ldots, N$ users. Thus for T time slots, there are 2^{NT} possible combinations. Hence the problem has an overall search complexity of order $O(2^{NT})$, which is computationally prohibitive.

Theorem 1 (NP-completeness)**.** *The user-slot assignment problem as described in Equation 3.1 is an NP-complete problem.*

Proof. Please refer Appendix A for the proof. ■

Chapter 4

Algorithm

Once we define our original problem as discrete optimization over a search space, we need to design an efficient algorithm to solve it. While there are several generic approaches to solve such problems, ranging from exhaustive search to stochastic local methods (such as genetic algorithm or tabu search), we propose the use of an A^* based search algorithm that is extensively used in path finding [12].

4.1 Motivation

The A^* algorithm [12] explores the search space in a systematic best first manner, using a heuristic to guide the search. A^* being a systematic search approach provides some advantages over other search methods. First, it can guarantee optimal solution which is generally not possible by non-systematic algorithms like genetic algorithm or tabu search. Second, it also guarantees that no other optimal algorithm can improve upon its performance (in terms of number of search steps). Also, A^* can be easily modified to an approximate algorithm (weighted A^*) using a weighted heuristic; weighted A^* guarantees bounded sub optimal termination and in practice is orders of magnitude faster than A^*, which makes it very useful for solving large scale minimization problems within reasonable resource constraints.

4.2 The A^* algorithm

In our problem, the search space is the set of all possible ways of filling the $N \times T$ association matrix $\mathbf{A}$ with 0's and 1's. The A^* algorithm explores the search space in a systematic manner, starting with an empty matrix and filling in columns as it progresses. Each filled column represents a time slot allocated —users assigned 1's being active in the time slot, while users assigned 0's are disabled. We denote any partially or completely filled matrix as a *state*. A state q with t filled columns thus represents a partial user-slot assignment in which we have so far covered t time slots (note that there are 2^{Nt} ways of doing so; one state represents only one such

way), while the remaining slots are yet to be allocated (refer Fig 4.1).

Users↓ \ Slots→	1	2	3	4	...	...	T
User 1	1	1					
User 2	0	0					
User 3	1	0					
User 4	0	0					

Figure 4.1: Illustration of a state q for a user-assignment problem comprising 4 users and T time slots. This state represents a partially filled association matrix in which $t = 2$ time slots have been allocated. A "1" for any user-slot combination implies that the user will be allowed to transmit in that time slot, whereas a "0" means that the user will remain disabled during the slot.

We now evaluate the cost of state q. Let $g(q)$ denote the cost of the assignments already made, which in our case is the total number of 1's in the t filled columns. Corresponding to the assignments in the first t columns, each user achieves certain data rate. If the total target rates of the users are not achieved, further assignments need to be made to the remaining time slots. We define $h(q)$, also called a heuristic, as an estimate of the minimum cost $h^*(q)$ of the remaining assignments at state q, i.e., the minimum number of 1's to be filled in the remaining columns. A heuristic is said to be *admissible* if $h(q) \leq h^*(q)$, i.e. if the heuristic function is a lower bound to the cost of the optimal assignment pattern for the remaining columns (A method for obtaining an admissible $h(q)$ is described later in Sec 4.3). We define $f(q)$ as the overall cost of state q as follows

$$f(q) = g(q) + h(q) \tag{4.1}$$

A^* search guided by the above cost function with an admissible heuristic is guaranteed to find the optimal assignment pattern satisfying Problem 1 [12].

The algorithm begins by filling the first column of the matrix **A** in all possible ways, thereby generating the first set of states. These states are inserted in an *OPEN* list —it is the list of states that are to be considered for further evaluation. From the *OPEN* list, the best state so far, i.e. the state with the lowest cost (computed using eq (4.1)), is chosen for expansion. When a state is expanded, it is removed from the *OPEN* list, child states are generated corresponding to all possible assignments for the next unfilled column, and are inserted in *OPEN*. As more states are expanded, the *OPEN* list comes to include states with more filled columns, so the estimate of the minimum cost keeps getting more accurate. This process of expanding the minimum cost states is continued until a state satisfying the target rate of all the users is found to have the minimum cost. Given that an admissible heuristic is used, the state thus obtained is the optimal solution to Problem 1.

4.2.1 Weighted A^* algorithm

The term "weighted A^*" encompasses several approximate variants of the A^* algorithm (which itself is optimal), that are faster than A^* and provide provable upper-bounds on sub-optimality. The version discussed in our work uses a constant weighting factor ρ to inflate the heuristic

$h(q)$ [17], unlike other methods that use dynamic weighting. This constant-inflation A^* search is often denoted as WA^* to distinguish it from other weighted A^* algorithms [9]. In our work, we use the generic term "weighted-A^*" to refer to the particular case of constant inflation.

With an inflation by a factor of ρ, the cost function in (4.1) is modified to:

$$f(q) = g(q) + \rho h(q), \qquad \text{with } \rho \geq 1 \tag{4.2}$$

This leads to the states with higher $h(q)$ being penalized, thus making the search greedier and often faster. The reduced computational time comes at the price of optimality. The weighted A^* algorithm is ρ-admissible, meaning that, given an admissible heuristic $h(q)$, the solution returned by weighted A^* has a cost that does not exceed the optimal cost by more than a factor of ρ [16]. However, in practice, the solutions obtained are better than what is guaranteed by the bounds [23]. For $\rho = 1$, the search is optimal, as ρ increases the search becomes faster but further from optimal.

4.3 Admissible heuristic for A^*

The admissible heuristic of the A^* is a lower bound on the cost of the optimal assignment for the unfilled columns. A heuristic which achieves a tight lower bound leads to a more efficient search. While there are several possible ways of obtaining admissible heuristics, we solve a relaxed version of the original assignment problem, to obtain a tight lower bound. The heuristic $h(q)$ at state q is obtained by solving the following problem

***Problem 2*:**

$$\begin{aligned} \text{Minimize } & \sum_{i=1}^{N} \sum_{j=t+1}^{T} a_{i,j} \\ \text{Subject to } & \sum_{i} R_i^{total} \geq \sum_{i} R_i^{target} \\ R_i(S,p) &= \frac{1}{|S|} \times \max_{l:|S_l|=|S|} \left(\sum_{i \in S_l} R_i(S_l, p) \right) \end{aligned} \tag{4.3}$$

The objective of Problem 2 is to minimize the total number of 1's in the remaining $(T-t)$ time slots, keeping the assignments of the previous t columns fixed, such that the sum of rates obtained over all lines conforms to the total target data rate of all lines. The second constraint in eq.(4.3) ensures that the sum-rate (sum over all lines) achieved by every n-tuple of users ($n = |S|$) in a time slot is the same, and is equal to the maximum sum-rate that can be achieved in a time slot by any n-tuple of users. We denote this value by $r_{|n|}$. The input sum-rate vector for Problem 2 is then $\{r_{|1|}, r_{|2|}, \ldots, r_{|N|}\}$, denoting the maximum sum-rates achievable in a time slot by 1,2,...,N active users respectively. With these constraints, Optimization problem 2 is simply to minimize the total number of 1's in the remaining $(T-t)$ time slots by deciding on the number of active users in each slot, such that the sum of gains achieved over these slots

complies with the remaining cumulative requirement of users. The total number of 1's in the obtained assignment pattern is the heuristic $h(q)$.

Theorem 2. *$h(q)$ obtained from solving problem 2 is an admissible heuristic.*

Proof. Let $h^*(q)$ be the minimum cost incurred in filling the remaining columns at state q while satisfying each user's pending rate requirement ($R_i^{total} \geq R_i^{target}, \ \forall i$). Let $h_1^*(q)$ denote the minimum cost of satisfying the corresponding sum-rate requirements ($\sum_i R_i^{total} \geq \sum_i R_i^{target}$). By collapsing all the users' constraints to a single constraint, we are expanding the feasible set of the original problem, hence $h_1^*(q) \leq h^*(q)$. Since the max-rate assumption in Problem 2 over-estimates the per-symbol rates of users, the target sum-rate can be achieved in fewer time slots. It follows that $h(q) \leq h_1^*(q) \leq h^*(q)$. ■

We now look for an efficient method to solve Problem 2. Given a state q with t filled slots, let R denote the minimum sum-rate that must be achieved in the remaining $(T-t)$ time slots. The objective is to find the minimum cost, i.e. the minimum number of 1's that must be assigned in the remaining columns in order to achieve sum-rate R. To find the minimum cost, we first solve a related problem of finding maximum sum-rate that can be achieved given a fixed value of c, $\forall c \in [0, N(T-t)]$. The minimum value of c at which sum-rate R is obtained in $(T-t)$ slots is the desired heuristic $h(q)$.

Now, the problem of determining the maximum achievable sum-rate in $(T-t)$ time slots given a cost c exhibits optimal substructure - one can build an optimal solution to the original problem from the optimal solution of subproblems. In other words, one can apply Dynamic programming to solve Problem 2. The following lemma demonstrates this fact.

Lemma 1. *Suppose we know that optimal solution of c-cost problem has u active users in the $(t+1)^{th}$ slot, then the optimal solution consists of an optimal solution to the remaining $(c-u)$ cost problem in $(T-t-1)$ slots, plus u active users in the $(t+1)^{th}$ slot [5].*

Let $\Delta R(T-t, c)$ denote the maximum sum-rate that can be achieved in $T-t$ slots, given cost c. Using Lemma 2, it can be obtained with the help of the following recursion

$$\Delta R(T-t, c) = \max_{n \in [0,N]} \{\Delta R(T-t-1, c-n) + r_{|n|}\} \tag{4.4}$$

where $r_{|n|}$ is the sum-rate achieved in the $(t+1)^{th}$ time slot for any n-tuple of users (recall the definition of $r_{|n|}$ following eq.(4.3)). Through the recursion in eq.(4.4), one can reach the optimal $\Delta R(T-t, c)$ using optimal $\Delta R(T-t-1, c-n)$ by searching over all possible number of active users ($n \in 1, \ldots, N$) in the $(t+1)^{th}$ slot.

4.4 An example

Fig 4.2 is an illustrative representation of the algorithm using a simple example for 3 users and 4 time slots. The association matrix is initially empty —this is depicted by the state q_0. Since q_0 is the only state in *OPEN*, it is picked for expansion. Expansion generates 7 new states (2^3 states, minus the state with null assignments, which is same as q_0). Each newly generated state is associated with an f-value as per eq (4.2) and inserted in *OPEN*. Assuming that q_4 is the state with the lowest f-value in *OPEN* after the update, it is selected for further expansion and removed from *OPEN*, while the others remain. Expansion of q_4 keeps the first column fixed, and fills in the second column with 0's and 1's. All the states thus generated are inserted in *OPEN*, which now contains the states generated from the expansion of q_0 as well as q_4. Again, the state with the lowest f-value is chosen from amongst all these states and expanded. The process continues until the topmost state in *OPEN* (the one with the lowest f-value) satisfies

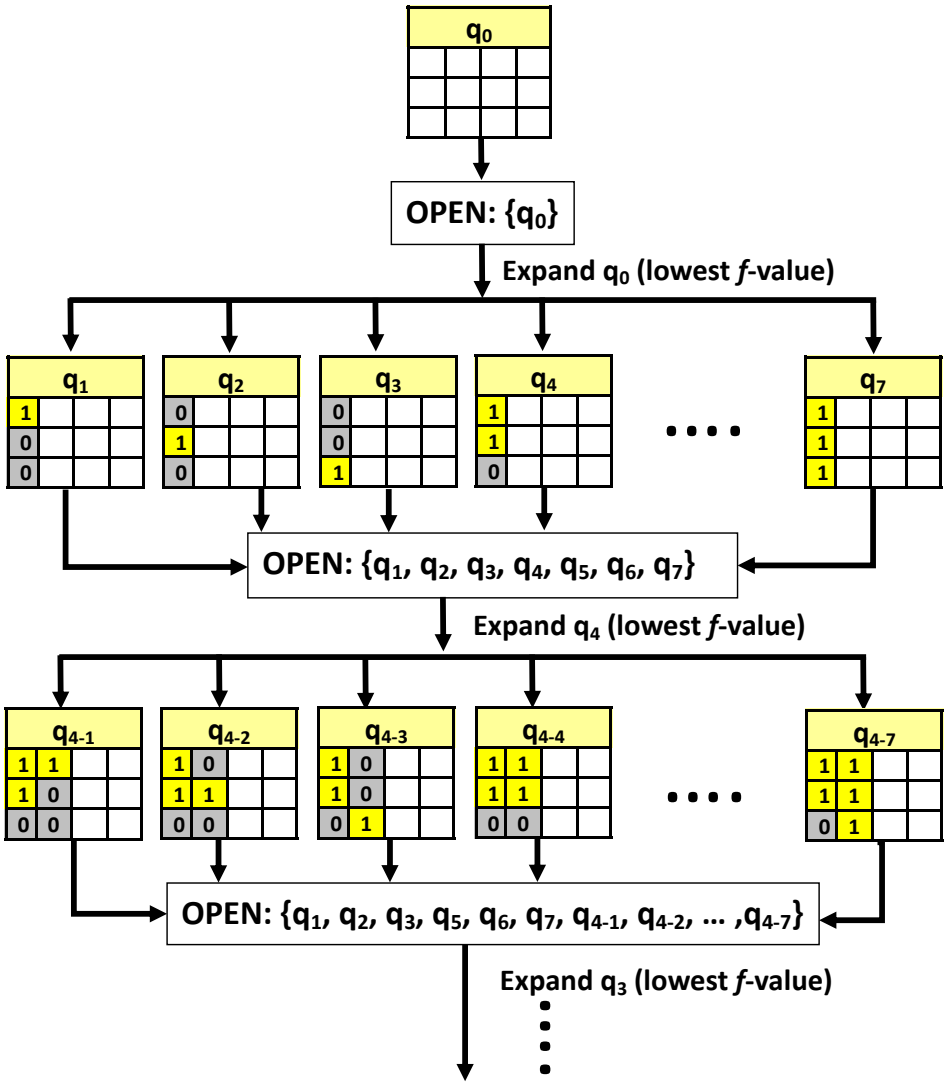


Figure 4.2: Illustration of the A^* algorithm for 3 users and 4 time slots

all the users' constraints.

4.5 Reducing the computational complexity

Ideally, the process of expansion of a state results in the generation of 2^N child states, corresponding to the 2^N unique user-combinations in a column. Considering only the combinations with n active users ($1 \leq n \leq N$), one may observe that the child states resulting from all these combinations have the same value of $g(q)$ as they have the identical number of 1's. These states differ only in the value of the heuristic $h(q)$ which depends upon the remaining rate requirements of users after filling the current column, which in turn depends upon the choice of active users in the column. To lessen the computational burden, we pick only one state corresponding to each value of $g(q)$. The selection of active users is done on the basis of the rate requirements of users prior to filling the current column, with users having higher rate requirements getting higher priority. This introduces more sub-optimality to the resulting solution because only N out of 2^N child states are inserted into the *OPEN* list during each state expansion. The search however becomes much faster, and acceptable user-slot assignment patterns can still be obtained, as shown in the results section.

Chapter 5

Simulation Setup

We perform experiments on G.fast field measurement data to evaluate the energy efficiency of the proposed user-slot assignment approach when employed parallel to the precoding-cum-bit-loading strategies discussed in Section 2.3, and compare with other standard precoding methods *viz.* full vectoring, optimal vectoring and discontinuous vectoring [14]. Note that *full vectoring* refers to the method where all the users (active as well as discontinued) are precoded for all used time slots, whereas *optimal vectoring* involves updating the precoder at every time slot to match the inverse of the crosstalk sub-matrix for the active users.

5.1 Topology

Simulations are performed on the G.fast 106 MHz profile for four different binder configurations. Binders 1 and 2 comprise of 10 and 16 lines respectively with loop lengths of 100m each; the crosstalk data for these binders has been obtained from the cable measurements provided by British Telecom. Binders 3 and 4 comprise lines with unequal loop lengths —Binder 3 consists of 12 lines with loop lengths of 100m to 200m in steps of 50m (4 lines at each loop length); Binder 4 consists of 10 lines with loop lengths from 100m to 300m in steps of 50m (2 lines at each loop length). The channel data for Binders 3 and 4 has been provided by Swisscom.

Transmit PSD is kept below -60dBm/Hz for frequencies below 30 MHz, and below -75dBm/Hz for frequencies above 30 MHz. Background noise is AWGN with a PSD of -140 dBm/Hz.

5.2 Traffic patterns

The binders are subjected to two types of traffic patterns. In the first traffic pattern (TP1), all users in a binder have equal rate requirements, ranging from 20 Mb/s to the highest rate achievable with vectoring. The second traffic pattern (TP2) is a light usage scenario, where the users with the highest and lowest aggregate channel capacity have a varying target data rate, whereas all the other users in the binder have a fixed minimal target of 200 Mb/s, since G.fast

systems are expected to sustain a minimum rate of 200 Mb/s for all users at all times [24].

5.3 Bit-loading and attainable data rates

The bit-loadings for all four binders, subject to different precoding strategies, are calculated from the crosstalk data using eq (2.3), with an efficiency of 0.85, SNR gap of 10.75 dB, and the maximum bits per bin limited to 12. Under certain precoding strategies, the bit-loadings remain fixed throughout all DMT symbol positions, in which case the net attainable data rates can be calculated using eq (2.4). In full vectoring, since all the users are precoded for all used time slots, the bit-loadings of active users remain same throughout. In optimal vectoring, bit loadings are assigned according to the worst case SNR over all possible user combinations. The bit-loadings for discontinuous vectoring are also fixed, calculated using the maximum residual noise over all user-combinations as described in [14].

Table 5.1: Maximum attainable data rates (in Mb/s) for Binder 4 under different precoding scenarios

	Loop length	Without crosstalk	No vectoring	Full vectoring	Optimal vectoring	Discontinuous vectoring	Discontinuous vectoring (variable bit-loading)
Line 1	300m	397	92	376	376	355	356
Line 2	300m	397	93	376	376	356	356
Line 3	250m	506	118	479	479	417	418
Line 4	250m	505	118	480	480	417	418
Line 5	200m	681	141	612	612	542	550
Line 6	200m	682	140	609	609	535	545
Line 7	150m	913	197	887	886	599	762
Line 8	150m	877	191	851	850	590	730
Line 9	100m	975	196	958	958	604	831
Line 10	100m	974	196	958	957	604	831

Table 5.1 shows how the data rates of users in Binder 4 vary as the precoding scheme changes. Columns 3 and 4 list the rates of users in the absence of crosstalk, and when crosstalk exists among all users but no precoder is employed. Columns 5-7 list the maximum attainable under three different scenarios —Full vectoring, Optimal vectoring and Discontinuous vectoring —with constant bit-loadings. The table indicates that full vectoring achieves data rates very close to the direct channel data rates in absence of crosstalk. Since the rates for optimal vectoring are calculated according to the lowest SNR for each user which usually corresponds to the case of all users being simultaneously active, they are almost same as those for full vectoring. The rates achieved due to discontinuous vectoring vary with loop length, being as high as 94% of the rates due to full vectoring for loop lengths of 300m, and as low as 63% of the rates due to full vectoring for loop lengths of 100m.

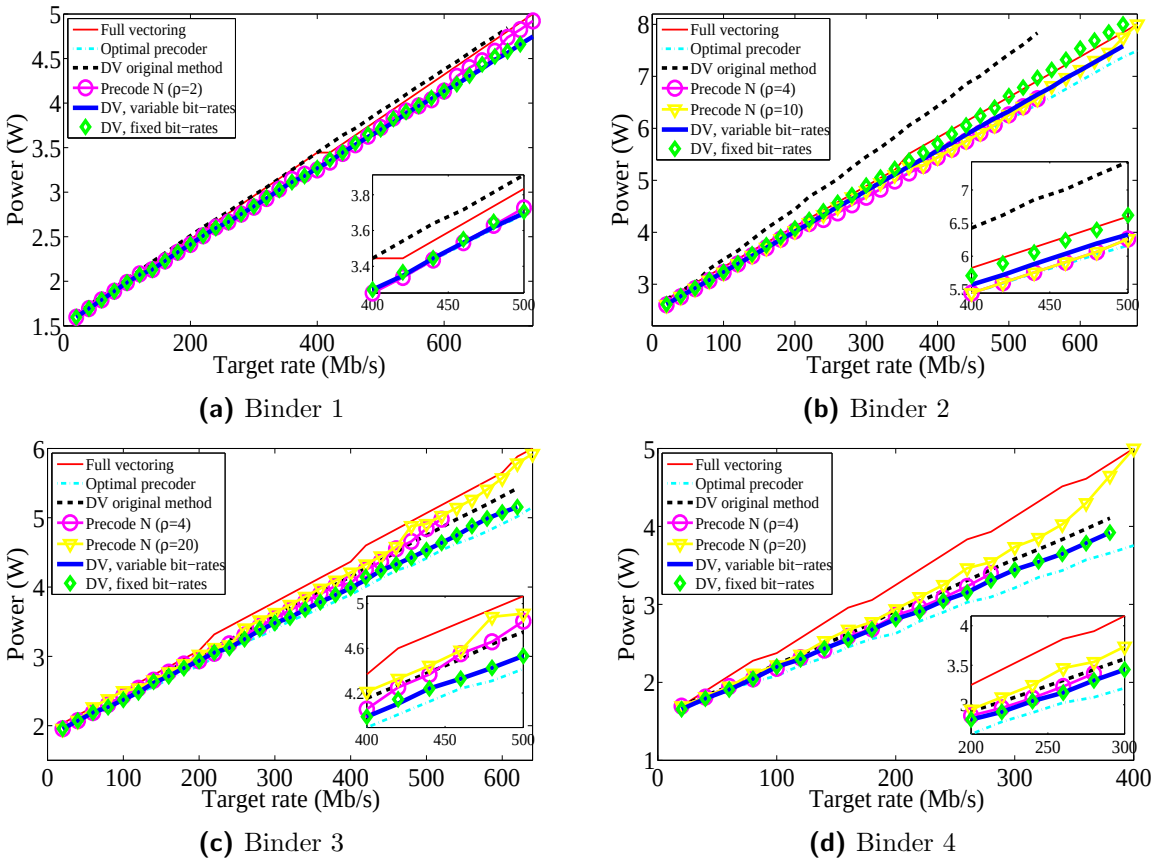
For the first two of the proposed precoding strategies in Section 2.3, bit-loadings are allowed to vary across time slots, thus maximum data rate for any user will be achieved when the user-

combination with the highest bit-loading for the user is repeated across all time slots in a given TDD frame. Note that the data rates thus obtained might not be simultaneously attainable for all users. Thus for the *Precode-N* scheme, the maximum achievable data rate for each user is same as the direct channel rate for the user (column 3 of Table 5.1), however it corresponds to the very rare scenario where only a single user transmits for an entire frame. A more feasible upper limit is the one obtained when all users transmit together in all time slots, the rates then correspond to the rates for full vectoring (column 5 of Table 5.1). For the second proposed scheme, DV with variable bit-loadings, the maximum achievable rates are listed in the last column of Table 5.1. Clearly, for certain users, allowing the bit-loadings to vary could lead to achieving potentially higher data rates than the original discontinuous vectoring method. For the third scheme, although the bit-loadings are fixed a-priori for each TDD frame, these values are a function of the data rate requirements of the users and are estimated after performing user-slot assignment. Hence the maximum attainable rates in this case have not been reported.

Chapter 6

Results

6.1 Comparison of Energy efficiency



(a) Binder 1 **(b)** Binder 2

(c) Binder 3 **(d)** Binder 4

Figure 6.1: Analog power consumption for all four binders, as a function of the target data rate, subject to Traffic Pattern 1, under different precoding strategies. Values on the x-axis depict the common target for each user in the binder. Values on y-axis depict average power consumption of the binder in a single DMT symbol position.

Fig 6.1 shows the power consumption of different precoding schemes for the uniform traffic pattern (TP1). The optimal precoding strategy, though practically infeasible in terms of im-

plementation, has the lowest power consumption since the ideal linear precoder for each user combination is employed.

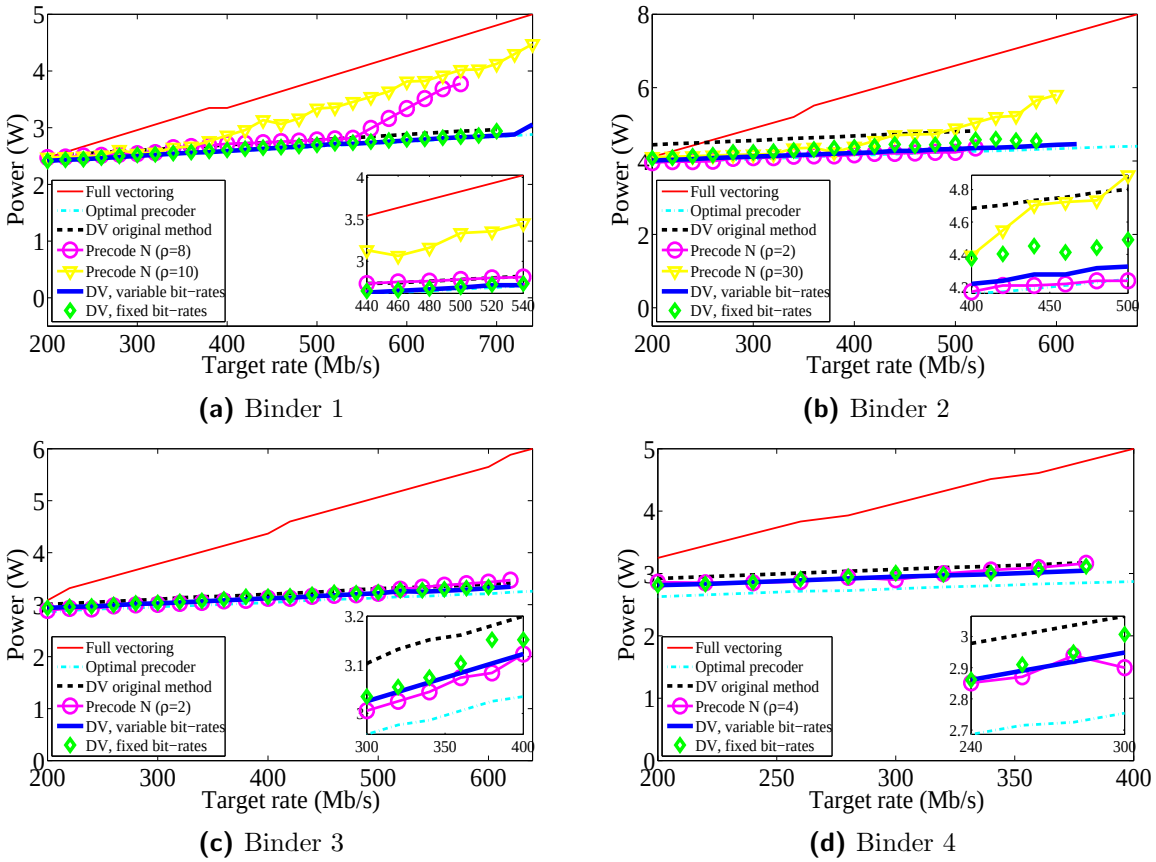
Power consumption of the full vectoring scheme is very close to optimal vectoring for Binders 1 and 2 comprising equal loop lengths. In this case, since all users have similar data rates (owing to equal loop lengths) and the same target rates (owing to the uniform traffic pattern), each user needs nearly the same number of symbol positions to meet its target, hence precoding all users is a profitable option. This does not hold for Binders 3 and 4 (Fig.6.1c and Fig.6.1d), which comprise mixed loop lengths, so that some users need to transmit for more symbol positions than others. Precoding all users when only a few are transmitting leads to idle symbols being transmitted on the discontinued lines as well. This explains the high power consumption of full vectoring for the mixed-length topologies.

The chief advantage of discontinuous vectoring over full vectoring is due to its ability to turn off analog components for discontinued lines [14]. This explains the much lowered power consumption for Binders 3 and 4 (Fig.6.1c and Fig.6.1d). However, for binders comprising lines with equal loop lengths, the lowered per-symbol capacities due to the residual noise introduced by discontinuous vectoring outweigh the power saved in a few symbol positions, thus leading to an overall power consumption higher than full vectoring (Fig.6.1a and Fig.6.1b).

The first of the three proposed precoding-cum-bit-allocation schemes, Precode-N, has a power consumption close to optimal vectoring, and considerably lower than discontinuous vectoring for the equal loop length binders (Fig.6.1a and Fig.6.1b). For mixed-length binders (Fig.6.1c and Fig.6.1d), its power consumption is at par with discontinuous vectoring. Note that the Precode-N scheme requires user-slot allocation, which is performed using the weighted A^* algorithm. As mentioned earlier, varying the value of ρ changes the optimality and computational time of the algorithm. Fig 6.1d shows the impact of varying ρ; for a lower value ($\rho = 4$), the computational time is large, hence the power consumption for the higher target rates have not been reported; for a higher value ($\rho = 20$), although the computation becomes faster, the overall power consumption goes up. This simple precoding scheme thus fares pretty well compared to the comparatively complex discontinuous vectoring.

Applying the user-slot assignment on top of discontinuous vectoring (thereby allowing bit-loadings to change over time slots) results in more power savings compared to the original DV scheme, as shown by the plots. The average power saving over all the target data points shown in the four plots in Fig 6.1 is 5.35%. The third proposed scheme, where the bit-rates are fixed for a time-frame, but calculated according to the rate requirements in the frame, leads to power consumption somewhere between the original DV scheme and DV with variable bit-rates. For this scheme, the average reduction in power consumption over the original DV scheme is 4.87%.

Fig 6.2 shows the power consumption for the non-uniform traffic pattern (TP2). In this case too, the optimal precoding strategy has the lowest power consumption of all the compared schemes. Power consumption for the full vectoring scheme is very high for all the binders, i.e. for both



(a) Binder 1 **(b)** Binder 2

(c) Binder 3 **(d)** Binder 4

Figure 6.2: Analog power consumption for all four binders, as a function of the target data rate, subject to Traffic Pattern 2, under different precoding strategies. Values on the x-axis depict the common target for the users with the highest and the lowest per-symbol capacities, while the rest of the users in the binder have a requirement of 200 Mbps. Values on y-axis depict average power consumption of the binder in a single DMT symbol position.

fixed-length and mixed-length topologies. This is because, owing to the non-uniformity in rate requirements of users, the number of active symbol positions for the users are never the same, so that there are several symbol positions where all users do not remain active simultaneously and hence precoding them all leads to unnecessary power consumption. Discontinuous vectoring in such scenarios performs much better than full vectoring.

For the Precode-N scheme, the power consumption for two different values of ρ have been reported for Binders 1 and 2 (Fig.6.2a and Fig.6.2b). This is to show that, using a higher value of ρ, power consumption corresponding to almost all target data points can be feasibly computed, while a lower value leads to considerably lower power consumption, although the computation becomes infeasible for higher target rates. The computational time requirements are discussed later in section 6.2. The plots show that power consumption using the lower ρ-value is comparable to or even less than that for DV scheme. Discontinuous vectoring with variable bit-rates improves upon the original discontinuous vectoring by an average drop of 4.86% (over all the target data points shown in Fig 6.2) in the analog power consumption. For Discontinuous vectoring with frame-wise fixed bit-loadings, the average drop in power consumption over original

Discontinuous vectoring is 4.73%.

Tables 6.1 and 6.2 show a comparison of the energy efficiency of the precoding schemes. Since optimal precoding has the lowest analog power consumption, we take it as a benchmark and calculate the percentage increase in power consumption for the other precoding schemes over optimal precoding, averaged over those target data rates that are attainable by each of the precoding strategies. We observe that in most cases employing a simple precoder only in the normal operation interval (Precode-N) along with a fairly efficient user-slot allocation is as good as, or even better than performing discontinuous vectoring. Although in some scenarios Precode-N does incur higher power consumption than DV, it is also capable of achieving higher target rates than DV does. Also, power consumption of DV reduces by a considerable margin by integrating user-slot allocation with DV. Moreover, user-allocation done on top of DV while still keeping the bit-rates fixed for the duration of a TDD frame (the third precoding scheme) also incurs lower power than the original DV method.

Table 6.1: Mean increase (in per cent) in power consumption over optimal precoding for TP1

	Full vectoring	DV original method	Precode-N	DV variable bit-rates	DV fixed rates per frame
Binder 1	3.29	4.71	0.59	0.13	0.32
Binder 2	4.95	13.97	1.01	1.66	3.61
Binder 3	10.77	5.21	7.15	1.92	2.01
Binder 4	20.88	8.15	11.13	4.52	4.53

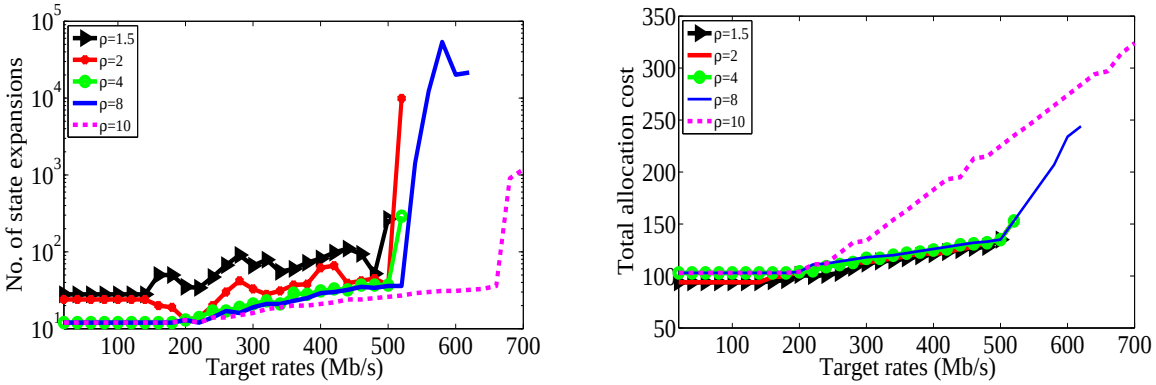
Table 6.2: Mean increase (in per cent) in power consumption over optimal precoding for TP2

	Full vectoring	DV original method	Precode-N	DV variable bit-rates	DV fixed rates per frame
Binder 1	38.50	4.27	19.95	0.40	1.00
Binder 2	33.51	12.65	0.41	1.63	4.87
Binder 3	47.96	5.24	3.18	2.64	3.28
Binder 4	49.77	11.09	7.69	8.10	7.75

We also note that, due to the lowered per symbol capacities resulting from the residual noise introduced, the original discontinuous vectoring method fails to achieve higher target rates, which can otherwise be met by full vectoring. This drawback is overcome by the Precode-N scheme, which can achieve target rates as high as those met by full vectoring. Although the computational complexity increases for high targets, it has been addressed by increasing the relaxation factor (ρ) in the A^* algorithm.

6.2 Time complexity of the weighted A^* algorithm

Section 4.2.1 mentions that increasing the relaxation factor ρ in the weighted A^* algorithm speeds up the search while sacrificing the optimality of the solution obtained. However, unlike the bound suggests, the decrease in solution quality is often not linear with the increase in weight [23]. Here, we examine the relation between the weight ρ and the number of state expansions in A^* (which determines the run time of the algorithm) for our problem domain.



(a) Number of state expansions

(b) Total allocation cost

Figure 6.3: Impact of varying the relaxation factor (ρ) of the weighted A^* algorithm while obtaining the user-slot assignment pattern for Binder 1 using the Precode-N strategy and under the non-uniform traffic pattern (TP2)

Fig 6.3 presents a comparison of the number of state expansions and the cost of the corresponding solutions obtained for five different values of ρ for a particular user-assignment problem. Fig 6.3a shows that increasing ρ monotonically reduces the number of state expansions. The number of state expansions also increases for the higher target rates. Data points which require more than 100 seconds for computation have not been shown in the figure. With this limitation, only $\rho = 10$ manages to obtain a solution for the highest target rate. Fig 6.3b shows the cost of the corresponding solutions obtained. There is little variation in the quality of solution obtained for the lower values of ρ ($\rho \leq 8$). However, for $\rho = 10$, the cost greatly spikes up, showing that the solution is farther from optimal.

The figures verify the claim that the quality of solution does not decrease linearly with the increase in ρ.

Part II

Resource allocation in WiFi

Chapter 7

Introduction

Rapid improvements in the physical layer bit rates supported by wireless links in 802.11 wireless LANs (WiFi) have been accompanied by users' demands for high throughputs (bits/sec/Hz/user). By employing a carrier sense mechanism, WLAN systems avoid simultaneous transmission of links connected to different access points that are within carrier sense range of each other. Due to this, sharing of spatial resource between such links remains inefficient. In [19], authors show that in networks with high client and access point densities, links could gain from transmitting simultaneously. While keeping access mechanisms as they are in standard 802.11 DCF[1], once access to transmit data has been gained by a link using RTS/CTS[2], other links are allowed to piggyback on the data transmission opportunity created and transmit their data simultaneously with the link. Although simultaneous transmission increases interference, the links get to transmit for larger time intervals and over higher bandwidths, hence throughput increases [18], [19].

7.1 Network model and throughput calculation

Consider a network comprising M Access Points (AP's) and N clients. All nodes (AP's and clients) are within each other's carrier sense range, and they access the medium using the DCF with RTS/CTS. Assume that all the links in the network are saturated, i.e. they always have a packet to send [19].

We first discuss the throughput calculation under standard DCF operation, details of which can be found in [4]. Successful transmission of data requires a link to reserve the medium by sending RTS. If no other links start transmitting when the RTS is being sent, the medium is successfully reserved. Let τ_s denote the probability of successfully sending an RTS. Let SNR_i be the signal-to-noise ratio of link i when it transmits exclusively over its data transmission opportunity. Its physical rate of transmission is given by $R_i = \log_2(1 + \mathrm{SNR}_i)$. Assuming that

[1]Distribution Coordination Function (DCF) is the primary medium access control (MAC) technique of the IEEE based 802.11 WLAN standard [4]

[2]Request-To-Send/Clear-To-Send is an optional handshaking technique used by the 802.11 wireless protocol to combat the problem of hidden terminals [4]

$\mathcal{T}_i$ is the time interval over which link i sends data, and the bandwidth is 1 Hz, the data payload, i.e., the number of bits sent by i in an average slot is given by [19]

$$\mathcal{D}_i = \tau_s \mathcal{T}_i \log_2(1 + \text{SNR}_i) \tag{7.1}$$

If σ denotes the length of an average slot, the corresponding throughput (bits/sec) is given by

$$T_i = \mathcal{D}_i / \sigma \tag{7.2}$$

Under the modified DCF proposed in [19], the network is split into share-in-space (SS) sets, which are sets of links that transmit simultaneously. Consider a link i belonging to an SS set S of size $|S|$. Since it gets to piggyback on the data transmission opportunities of every other link in the set S, its probability of getting a successful transmission in $|S|\tau_s$. However, link i will also experience interference from all the other link in the set S, hence it will have a signal-to-interference-and-noise ratio, say SINR_i, which is lower than SNR_i. The average data payload of link i over an average slot of the modified DCF operation is thus

$$\mathcal{D}_i^{(S)} = \tau_s |S| \mathcal{T}_i \log_2(1 + \text{SINR}_i^{(S)}) \tag{7.3}$$

Eqs (7.1) and (7.3) imply that a gain in payload and hence throughput can only be achieved if the spatial multiplexing gain of $|S|$ outdoes the reduction in rates due to the interference introduced. Authors in [19] seek to partition the network into SS sets such that the overall network throughput is maximized.

7.2 Optimization Problem

Let $\mathcal{N} = 1, 2, \ldots, N$ be the set of links in a network. A partition of $\mathcal{N}$ is a set of mutually exclusive and collectively exhaustive SS sets. If the cumulative throughput of all links in an SS set p is denoted by T_p, then the overall throughput of the network partitioned as P will be given by $T_P = \sum_{p \in P} T_p$. Define $\mathcal{P}$ to be the set of all partitions of the network. Then the optimization problem of finding the partition that maximizes the network throughput is given by [19]

$$\begin{aligned} &\text{Maximize} \quad T_P \\ &\text{Subject to} \quad P \in \mathcal{P} \end{aligned} \tag{7.4}$$

This problem is NP hard in nature, hence authors in [19] have come up with heuristic methods to solve it. Authors introduce an upper bound on the network throughput found by solving a relaxed version of the original problem, denoted as Optimal Integer problem (OIP). Two computationally efficient algorithms —Reevaluate Optimal Integer Partition (ROIP) and Begin Sharing in Space (BSS) —have also been proposed that achieve a large fraction of the gains promised by the upper bound.

Chapter 8

Proposed approach

We re-look the network partitioning problem in eq (7.4) as a resource allocation problem. With N links in the network, transmission under standard DCF can be viewed as data transmission by N links in a time frame comprising N discrete time slots, with each link transmitting in exactly one slot. Each link has a throughput determined by its SNR, and the total network throughput is the sum of throughputs of links over the entire frame. With the modified DCF operation, multiple links may transmit simultaneously in one time slot (sharing in space), and links get the opportunity to transmit in more than one time slot (piggybacking on other links). The number of time slots a link gets to transmit for is equal to the number of links in the SS set of which it is a member. An important condition for simultaneous transmission is mutual exclusivity of SS sets, i.e., a link may share time slots only with a fixed set of links, it cannot be a part of multiple SS sets. Additionally, the SS sets must be collectively exhaustive, i.e., each link must be alloted a transmission opportunity.

With these insights, we frame the network partitioning problem as an equivalent resource allocation problem of filling an $N \times N$ association matrix $\mathbf{B}$ with 1's and 0's such that when link i transmits in j^{th} slot, $b_{i,j} = 1$, otherwise 0. Links sharing a time slot are members of an SS set. The aim is to maximize the network throughput, while ensuring that the above-mentioned conditions for feasibility of a network partition are satisfied.

For instance, consider a network comprising 5 links. A possible partition of the network is: { {1,3,4},{2,5} }. The corresponding association matrix $\mathbf{B}$ is shown in Fig 8.1a.

8.1 A^* algorithm

We now apply the modified version of weighted A^* algorithm suited for MAX problems [20], to solve the optimization problem at hand. Define a *state* as a collection of mutually exclusive SS sets, that need not be collectively exhaustive. In other words, a state is an incomplete partition of the network. It corresponds to a completely or partially filled association matrix, in which some columns are filled (corresponding to the time slots for the assigned SS sets), while the

Links↓ \ Slots→	1	2	3	4	5
Link 1	1	1	1	0	0
Link 2	0	0	0	1	1
Link 3	1	1	1	0	0
Link 4	1	1	1	0	0
Link 5	0	0	0	1	1

(a) Association matrix corresponding to a partition { {1,3,4},{2,5} } of a network comprising 5 links

Links↓ \ Slots→	1	2	3	4	5
Link 1	1	1	0		
Link 2	0	0	0		
Link 3	1	1	0		
Link 4	0	0	0		
Link 5	0	0	1		

(b) Incomplete association matrix for a *state*: corresponding to an incomplete partition { {1,3},{5} } of a network comprising 5 links

Figure 8.1

remaining slots are yet to be allocated (Refer Fig 8.1b).

Consider a state q in which n time slots have been allocated, i.e., the SS sets assigned so far consist of n links in total. Define $g(q)$ as the reward collected so far, which is nothing but the sum of throughputs of the SS sets assigned in n slots. Define $h(q)$ as an estimate of the remaining reward, i.e. of the maximum achievable throughput in the remaining time slots. The heuristic is said to be *admissible* if for every state q, it holds that $h(q)$ upper bounds the highest reward of the assignments for the unfilled slots in the state. For the weighted A^* algorithm with a relaxation factor of ρ, the total reward $f(q)$ of a state is defined as

$$f(q) = g(q) + \rho h(q), \quad \text{where } 0 \leq \rho \leq 1 \tag{8.1}$$

A^* is an iterative algorithm, starting the search with an empty state in which no SS sets are assigned, and *expanding* a single state in every iteration. Expanding a state q involves exploring all possibilities of adding another SS set to the state, thereby generating new states. Generated states are stored in a sorted *OPEN* list which is ordered by the reward function in eq (8.1). *OPEN* initially consists only of the empty start state. In every iteration, the state with the highest f-value in *OPEN* is expanded. As the search progresses, generated states enter *OPEN* and expanded states are removed from it. When a fully assigned state (comprising mutually exclusive and collectively exhaustive SS sets) is chosen for expansion, and it satisfies the condition that $h(q) = 0$, the search halts and the incumbent state is returned as the solution.

Theorem 3 (ρ-admissibility of weighted A^*)**.** *If $h(q)$ is admissible, the solution obtained using the above algorithm has a reward that is at least ρ times the highest reward.*

Proof. We quote Theorem 2 from [20] for the ρ-admissibility of weighted A^* in MAX problems.

> For any $0 \leq \rho \leq 1$, when the maximal f-value in *OPEN* is less than or equal to the reward of the incumbent solution C, then it is guaranteed that $C \geq \rho\, C^*$, where C^* denotes the cost of the optimal solution.

In our algorithm, an incumbent solution state q must satisfy the condition $h(q) = 0$. Hence $f(q) = g(q)$, i.e. the reward of the incumbent solution is same as its f-value. Now, $f(q)$ cannot

be less than the maximal f-value in *OPEN*, because q has already been chosen for expansion. Therefore, reward of the incumbent solution is greater than or equal to the maximal f-value in *OPEN*. Hence, by the above quoted theorem, $C \geq \rho\, C^*$ holds. ■

8.2 Admissible heuristic

An admissible heuristic for a state q should be an upper bound to the maximum achievable reward in the remaining time slots. We obtain the required heuristic from the Optimal Integer Partition (OIP) formulation in [19]. Given a network $\mathcal{N}$ with N nodes, OIP(N) provides an upper bound on the throughput of all partitions of the network. It is solved using a dynamic programming formulation (refer Sec IV in [19]), whereby OIP(k) is also computed ($\forall k \in [1, N]$), which gives an upper bound on the highest throughput achievable by any sub-network of $\mathcal{N}$ comprising k links. Thus for a state q where so far n links have been assigned to SS sets, an admissible heuristic for the remaining $(N - n)$ columns is obtained by solving OIP($N - n$).

Chapter 9

Results

9.1 Simulation Setup

Following the experimental setup in [19] and [18], networks comprising 15 clients distributed uniformly and independently over a two dimensional region are simulated. We consider different client densities, comprising a client every 2,4,8,16,32 and 50 m^2. The number of AP's is taken as 5, 10 or 15. For networks comprising equal number of clients and AP's, an AP is placed at an height of 2m above each client, and each bi-directional link has an SNR of 100. For networks with fewer AP's, clients are grouped into clusters and each cluster is assigned an AP. SNR of links is now a function of the distance between client and AP locations. Moreover, in these cases, an additional constraint on feasible network partitions is introduced; an SS set may contain only one node from each cluster. Wireless propagation is assumed to follow the pathloss model with a pathloss exponent $\alpha = 3$.

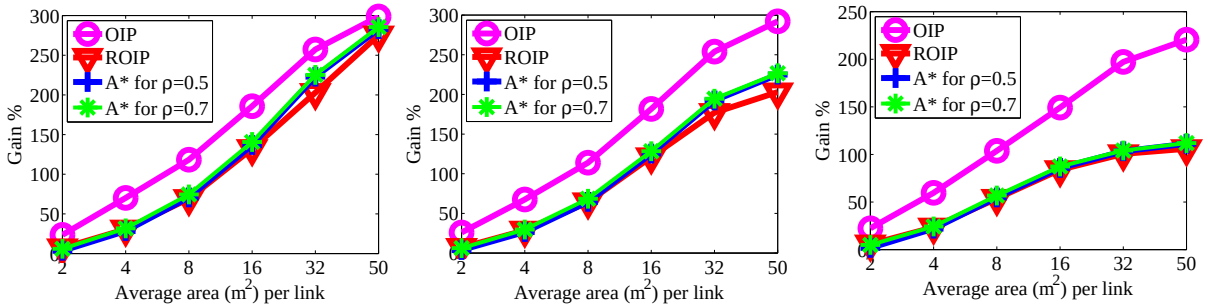
All results shown are for 50 network instances generated for each client/AP density combination.

9.2 Results

We use the weighted A^* algorithm with different values of ρ to derive efficient network partitions. The gain in throughput of the partitioned network under the modified DCF operation over the throughput achieved under standard DCF is computed. The results are compared with the ROIP method, and also with the upper bound on throughput gain obtained using OIP. Fig 9.1 shows the comparison. Significant gains over standard DCF operation are achieved. The upper bound obtained from OIP becomes loose as the number of AP's decreases. Gains obtained using weighted A^* are at par with those from ROIP, even higher in a few cases. For the two values of ρ used in the simulations, similar results are obtained.

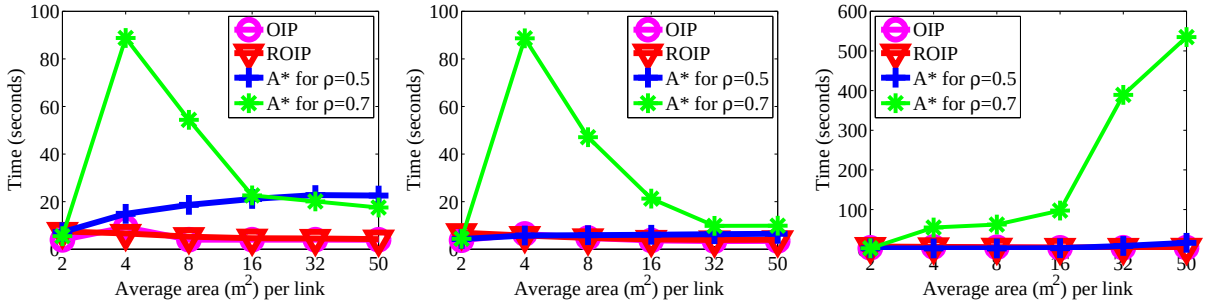
Fig 9.2 shows the timing information of the algorithms. Experiments were performed on a 2.0 GHz Intel Xeon x64 based system with 128 GB RAM. Note that OIP and ROIP have been run on 2 parallel cores, whereas A^* runs on a single core. A^* with $\rho = 0.7$ takes significantly higher

time than A^* with $\rho = 0.5$, although the throughput gains obtained using both are almost same. The run times of ROIP and A^* with $\rho = 0.5$ are of the same order.



(a) 15 clients and 15 AP's **(b)** 15 clients and 10 AP's **(c)** 15 clients and 5 AP's

Figure 9.1: Throughput gains obtained from OIP, ROIP and weighted A^* for different network configurations



(a) 15 clients and 15 AP's **(b)** 15 clients and 10 AP's **(c)** 15 clients and 5 AP's

Figure 9.2: Comparison of run times of OIP, ROIP and weighted A^* for different network configurations.

Chapter 10

Conclusion

We formulated the power minimization problem in G.fast as a user-slot assignment problem. We proved that it is an NP complete problem, and proposed a weighted A^* algorithm to obtain bounded suboptimal solutions to it. Admissible heuristics for the algorithm were obtained by solving relaxed versions of the original problem using a dynamic programming formulation.

We studied the impact of performing user-to-slot assignments in different precoding scenarios. Simulations show that the simple Precode-N scheme achieves reasonable energy efficiency when an efficient assignment strategy is employed. This scheme offers the convenience of precoding users only in the normal operation interval of a TDD frame, while still satisfying high data rate requirements of users. The ensuing power consumption was found to be less than that of the discontinuous vectoring scheme in several cases. We also applied our user-slot assignment on top of the discontinuous vectoring strategy, and found it that reduces the average power consumption of DV by 5.35% and 4.86% respectively for TP1 (uniform traffic pattern) and TP2 (non-uniform traffic pattern).

The proposed user-slot assignment strategy requires the bit-loadings of users to be modified in every time slot. This is achieved by communicating the active bit-loading tables for all time slots via the RMC symbol contained in every TDD frame.

We also addressed the optimization problem of finding a network partition for maximizing throughput by allowing simultaneous data transmissions in 802.11 systems. While authors in [19] found an upper bound to the maximum achievable throughput, and proposed heuristic based algorithms to approach the upper bound, we used the weighted A^* algorithm to solve the problem, that provides bounds on the quality of the solution obtained. Comparison of the methods shows that throughput gains using ROIP proposed in [19], and using A^* with $\rho = 0.7$ are very close. Theoretically, this implies that in most cases, network throughput obtained using ROIP is at least 0.7 times the optimum throughput.

Appendix A

Proof of Theorem 1

For the first part of the proof, we convert the slot assignment problem into a decision problem, i.e, for a given integer k, we ask whether the slot assignment problem can produce a solution such that $\sum_{i,j} a_{i,j} \leq k$. This decision problem is equivalent to the original problem as we know $0 \leq k \leq N \times T$, which means that we can solve the decision version of the problem at most $N \times T$ (polynomial in input) times to obtain the solution for the optimization version of the problem.

Next, we prove that this decision version of the problem lies in NP. This is fairly simple, as we need to show that given a solution for the decision version of the slot assignment problem, we can verify its correctness in polynomial time. Consider a given solution of this problem (decision version with a chosen k) as an $N \times T$ matrix. We can count the number of 1's in each column of the matrix to check whether the sum is $\leq k$, if not, the solution is incorrect. This iteration traverses the matrix cells at most once, and thus is polynomial. If the sum is $\leq k$, we can check row-wise for each user if their constraints are met (i.e, $\sum_{j:S_j \ni i} R_i(S_j, p) \geq R_i^{target}$), again if the constraints are not met by any of the user the solution is incorrect. Note that, this loop also traverses the matrix cells once only and thus is polynomial. Finally, if a solution is not proved incorrect in either of the above mentioned polynomial loops, it is correct. There, the slot assignment problem is an NP problem.

Next, we prove that the slot assignment problem is NP-hard. To prove this, we perform a reduction of the generalised cost bin packing problem (GCBP [3, 10]) to the slot assignment problem. GCBP is an NP-complete problem that can be seen as a generalization of the classical bin packing problem. The input for GCBP consists of a set of n items $I = \{1, 2, \ldots, n\}$ with different sizes $\{s_1, s_2, \ldots, s_n\}$ where $\forall i, s_i \leq 1.0$ and a concave non decreasing function $f : 0, 1, 2, \ldots, n \rightarrow \mathbb{R}$ (with $f(0) = 0$). The optimization goal is to partition the n items in m bins $\{B_1, B_2, \ldots, B_m\}$ such that $\sum_{j \in B_i} s_j \leq 1.0$ for any $1 \leq i \leq m$ and $\sum_{i=1}^{m} f(|B_i|)$ is minimized. In other words, the problem is to pack n items into m bins, where none of the bin's size constraint is exceeded while minimizing the sum of cost function that depends on the number of occupants of a bin.

Now given, such a GCBP we construct a time-slot assignment problem in the following way, each item is a user (i.e., n users) and each time slot is a bin (m bins). Now, we construct $R_i(S_j, p)$, which denotes the rate of user i when sharing a time-slot with user set S_j while adopting a precoding strategy p, in the following way,

$$\begin{aligned} R_i(S_j, p) &= 0 \quad \text{if } \textstyle\sum_{s_j \in S_j} s_j + s_i > 1.0 \text{ or } \sum_{k=1}^{m} a_{i,k} > 1 \\ &= s_i \quad \text{otherwise} \end{aligned} \tag{A.1}$$

The constraints in Equation (A) ensures i) the size of a bin is never exceeded and ii) an item is always placed in a single bin. Now, if we solve the time-slot assignment optimally, this will also be an optimal solution for GCBP as the $f(|B_i|), \forall i = 1, \dots m$ is non decreasing with $|B_i| = \sum_k a_{k,i}$. Therefore, the time-slot problem is an NP-hard problem, and as it is also an NP problem, it is NP-complete [1].

[1]Note that, GCBP imposes the constraint that each item is placed in one bin only, whereas for the time-slot assignment problem we partition the constraints of each user in addition to partitioning the users across slots, which makes it a more difficult optimization than GCBP.

Bibliography

[1] Accelerating Gigabit Broadband: A FTTdp White Paper. Tech. rep., Adtran, 2014.

[2] ITU-T Recommendation G.9701: Fast access to subscriber terminals (G.fast) - Physical layer specification, 2014.

[3] Anily, S., Bramel, J., and Simchi-Levi, D. Worst-case analysis of heuristics for the bin packing problem with general cost structures. *Operations research 42*, 2 (1994), 287–298.

[4] Bianchi, G. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE Journal on selected areas in communications 18*, 3 (2000), 535–547.

[5] Biyani, P., Prakriya, S., Bagchi, A., and Prasad, S. Dynamic programming based multi-user resource allocation for partial crosstalk cancellation in VDSL. *IEEE communications letters 16*, 3 (2012), 420–423.

[6] Brown, L. Overview of G.fast : Key functionalities and technical overview of draft Recommendations G.9700 and G.9701. G-fast summit 2014, 2014.

[7] Cendrillon, R., Moonen, M., Van den Bogaert, E., and Ginis, G. The linear zero-forcing crosstalk canceler is near-optimal in dsl channels. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE* (2004), vol. 4, IEEE, pp. 2334–2338.

[8] Ciofli, J. M. A Multicarrier Primer. *ANSI T1[E] 4* (1991).

[9] Ebendt, R., and Drechsler, R. Weighted A* search–unifying view and application. *Artificial Intelligence 173*, 14 (2009), 1310–1342.

[10] Epstein, L., and Levin, A. Bin packing with general cost structures. *Mathematical programming 132*, 1-2 (2012), 355–391.

[11] Ginis, G., and Cioffi, J. M. Vectored transmission for digital subscriber line systems. *IEEE Journal on selected areas in communications 20*, 5 (2002), 1085–1104.

[12] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 2 (July 1968), 100–107.

[13] Maes, J., Guenach, M., Hooghe, K., and Timmers, M. Pushing the limits of copper: Paving the road to FTTH. In *2012 IEEE International Conference on Communications (ICC)* (2012), IEEE, pp. 3149–3153.

[14] Maes, J., and Nuzman, C. Energy efficient discontinuous operation in vectored G.fast. In *2014 IEEE International Conference on Communications (ICC)* (June 2014), pp. 3854–3858.

[15] Müller, F. C., Lu, C., Eriksson, P.-E., Höst, S., and Klautau, A. Optimizing power normalization for G.fast linear precoder by linear programming. In *2014 IEEE International Conference on Communications (ICC)* (June 2014), pp. 4160–4165.

[16] Pearl, J. Heuristics: intelligent search strategies for computer problem solving.

[17] Pohl, I. Heuristic Search Viewed as Path Finding in a Graph. *Artif. Intell. 1*, 3 (1970), 193–204.

[18] Singh, M., and Kaul, S. On large throughputs in high density wireless local area networks.

[19] Singh, M., Kaul, S., and Biyani, P. On large throughputs in high density enterprise wireless lan (s). In *Global Communications Conference (GLOBECOM), 2014 IEEE* (2014), IEEE, pp. 4864–4869.

[20] Stern, R. T., Kiesel, S., Puzis, R., Felner, A., and Ruml, W. Max is more than min: Solving maximization problems with heuristic search. In *Seventh Annual Symposium on Combinatorial Search* (2014).

[21] Strobel, R. G.fast technology and the FTTdp network. Tech. rep., Lantiq, 10 2014.

[22] Strobel, R., Joham, M., and Utschick, W. Achievable rates with implementation limitations for G.fast-based hybrid copper/fiber networks. In *2015 IEEE International Conference on Communications (ICC)* (June 2015), pp. 958–963.

[23] Thayer, J. T., and Ruml, W. Faster than weighted a*: An optimistic approach to bounded suboptimal search. In *ICAPS* (2008), pp. 355–362.

[24] Timmers, M., Hooghe, K., Guenach, M., Stony, C., and Maes, J. System design of reverse-powered G. fast. In *2012 IEEE International Conference on Communications (ICC)* (2012), IEEE, pp. 6869–6873.

[25] Whiting, P., Ashikhmin, A., Borst, S., Jennen, J., Kramer, G., de Lind van Wijngaarden, A. J., and Zivkovic, M. Performance results for digital subscriber line precoders. *Bell Labs Technical Journal 13* (2008), 147–161.